WMX3

Technical Manual

Technical Support Guide

2021. 04 Revision 1.0

WMX3 Technical Manual

당사의 모션제어 솔루션 WMX3 제품을 선택해 주셔서 감사합니다. WMX3 Technical User Manual 을 통하여 WMX3 제품을 사용하기 앞서 아래 사항을 숙지하시어 제품을 올바르게 사용해 주시기 바랍니다.

- *사용설명서의 일부 또는 전부를 무단으로 복제하여 배포 및 사용할 수 없습니다.*
- *사용설명서의 내용은, 제품의 기능 향상을 위해 예고 없이 변경될 수 있습니다.*
- *매뉴얼의 내용 중 일부는 어플리케이션 버전에 따라 지원되지 않는 함수가 포함될 수도 있습니다.*
- *매뉴얼의 오류나 문의 사항에 대해서는 당사로 문의 주시기 바랍니다*

Information Company
25, Hwangsaeul-ro 258beon-gil, Bundang-gu,
Seongnam-si, Gyeonggi-do, Republic of Korea
TEL +82+31-895-5066 **FAX** +82+70-4015-5066

www.movensys.com

Movensys Co., Ltd.

Revision History

Ver.	*Date.*	*WMX3*	*RTX64*	*Summary*
1.0	21.04	3.4u4	3.7.3	초판

Information Company
25, Hwangsaeul-ro 258beon-gil, Bundang-gu,
Seongnam-si, Gyeonggi-do, Republic of Korea
TEL +82+31-895-5066 **FAX** +82+70-4015-5066

www.movensys.com

Movensys Co., Ltd.

Contents

1. WMX3

페이지 디자인 추가

1.1. 제품 개요

Movensys WMX3 는 반도체, FPD, 스마트폰 제조 장비 및 2 차 전지 등 제조 장비 분야의 풍부한 양산 경험을 기반으로 개발된 공장 자동화 설비에 적합한 새로운 모션 제어 솔루션입니다. 고속 시퀀스, PC 리소스 최적화, 프로그램과의 호환성, 프로그래밍 편의성 등 다양한 부분을 고려하여 설계된 새로운 소프트웨어 아키텍처로 범용 모션 제어 컨트롤러를 넘어 대용량 데이터 관리가 가능한 실시간 제어 및 관리 데이터 메모리를 제공합니다. WMX3 는 PC 를 활용한 소프트웨어 모션 기술로 구현되었기 때문에 필요한 데이터를 PC 메모리에서 실시간 관리가 가능합니다. 이에, 빅데이터 기반의 인공지능, 클라우드 컴퓨팅과 같이 많은 연산양을 필요로 하는 미래 산업 시대에 적합한 모션 제어 솔루션입니다.

WMX3 의 특별함

- 소프트웨어로 설계된 모션 제어 솔루션
- Windows PC에서 실시간 제어 기능
- EtherCAT Class A로 자체 개발한 EtherCAT 마스터
- 스마트한 모니터링 툴, 3D 시뮬레이터
- 2차 개발이 가능한 유연한 아키텍처
- WMX3 기능에 따른 다양한 라인업 구성

1.2. 제품 구성

WMX3 는 RTX (Real Time OS) 포함 패키지로 구성되어 있으며 라이선스에 따라서 접점제어를 위한 I/O 기능만 포함한 WMX3 I/O 버전, 모션 기능에 따른 WMX3 Standard, Advanced 버전으로 구성되어 있습니다.

RTX Real-time Windows IntervalZero WMX3 All-in-One Soft Motion Control EtherCAT

WMX3 기본 패키지

- Windows Library, 유틸리티, 샘플 코드, 매뉴얼 제공
- RTX64, RTX 관련 유틸리티, 매뉴얼 제공

	Function	*Summary*
WMX3 Standard	PTP (Point To Point) Motion Control	PTP 모션 제어 기능
	Linear | Circular | Helica | Interpolation	보간 제어
	Real-time Event Control	실시간 이벤트 제어
	Real-time Trigger Motion	실시간 트리거 제어
	Gantry Sync Control	겐트리 동기 제어
	Cyclic Sync Control	사용자 모션 프로파일 탑재 기능
WMX3 Advanced	Compensation	스테이지 기반 1D, 2D 위치 보상 기능
	Path Interpolation	경로 보간 제어
	Path Interpolation Look Ahead	선행 경로 보간 제어
	Spline	스플라인 기능
	PVT	PVT 기능
	E-CAM	고급 동기 기능
WMX3 I/O	I/O Control	I/O 접점제어를 위한 기능

1.3. 제품 사양

WMX3 사양은 아래와 같습니다. 구매하신 제품 라이선스 및 버전에 따라서 일부 사양 정보는 상이 할 수 있으니 제품 구매 시 반드시 라이선스, 버전 대비 사양을 체크하시기 바랍니다.

WMX3 Specifications	
제어 축수	4 ~ 128 축 | 동기, 보간 제어 시 최대 64 세트
I/O 점 수	Input 1 ~ 64000 | Output 1 ~ 64000 | Total 약 1.5KB
최대 노드	256
통신 주기(ms)	0.125(16 축) | 0.25(32 축) | 0.5(64 축) | 1(128 축) | 2 | 4
통신 플랫폼	EtherCAT | RTEX
통신 연결 방식	Line | Tree | Ring | Star
개발 언어	C | C++ | C# | Python | VB
개발 환경	Visual Studio 2012, 2013, 2015, 2017, 2019 | Labview | C++Builder

WMX3 Motion Specifications	
속도 커브	Trapezoidal | S-Curve | Jerk | Two-Velocity | Timed Profile
가속도 커브	Advanced Jerk | Sinusoidal | Parabolic | MAT
보간 기능	직선 | 원호 | 3D | 헬리컬 | PVT
궤적 기능	직선 및 원호 연속 궤적 | 스플라인 (B/C) 궤적 | Look-Ahead 속도 자동 조정
동기 기능	일반 동기 | 동기 편차 보상 | E-CAM
고속 기능	I/O 및 모션 조건에 따른 이벤트 등록 | 실시간 시퀀스 버퍼
보상 기능	Backlash 보상 | 3D 위치 보상

1.3.1. PC 권장 사양

WMX3 사용시 다음과 같은 PC 사양을 권장하며 이외 사양의 PC 는 반드시 검증이 필요합니다.

Hardware Specifications	
CPU	Intel 6 ~ 8 세대 i3, i5, i7
OS	Windows 10 PRO, LTSC
Win Ver.	Windows 10 1809, 1903, 1909, 2004
NIC	Intel I210, I211, I350
RAM	4GB 이상

Minimum Hardware Specifications	
CPU	Intel Atom 계열 | 2Ghz | Dual Core
기타	RAM 2GB | HDD/SDD 25GB

1.3.1.1. CPU

WMX3 제품 사용 시 필요한 CPU 기본 사양은 아래의 내용을 참고해주시기 바랍니다.

- Hyper Threading은 비활성화를 권장합니다.
- CPU 최소 성능은 적어도 Atom E3845 (2GHz, 2Core) 급 이상의 성능을 요구합니다.
- RTX 전용으로 코어를 하나 할당해야 하기 때문에 2개 이상의 코어를 필요로 합니다.
- Intel Skylake (6세대) 이후 세대의 CPU의 경우 오직 Windows 10 에서만 호환 가능합니다.
- Windows 7 SP1, Windows 8.1은 지원하지 않습니다.

*Note	
✓	Skylake 이후 세대의 CPU를 사용할 경우, 특정 RTX64 업데이트를 필요로 할 수 있습니다. 관련 자료는 아래의 Intervalzero RTX 참조 항목을 이용하시길 바랍니다.
✓	Hyper Threading이 켜진 상태라면, RTX를 통한 실시간성이 확보되지 않을 수 있습니다.

Intervalzero RTX Reference Link	
CPU	Intervalzero RTX64 Processor Compatibility
Hyper Threading	Intervalzero RTX Hyper-Threading Support
	Intervalzero RTX64 Hyper-Threading Support

1.3.1.2. OS

WMX3 는 아래의 윈도우 버전을 지원합니다. 지원하지 않는 윈도우 에서는 RTX 및 기타 문제가 발생할 수 있으므로 주의하시기 바랍니다.

	WMX3 32bit (x86)	WMX3 64bit (x64)
Windows OS	Windows 7 with SP1 (Recommended)	Windows 10 IoT Enterprise LTSC (Recommended)
	Windows 7 (No Service Pack)	Windows 10 (Ver. 1809, 1903, 1909, 2004)
	Windows Embedded Standard 7 with SP1	Windows 8.1 with Update
	Windows Embedded Standard 7 (No Service Pack)	Windows Embedded 8.1 with Update
		Windows 7 with SP1
		Windows Embedded Standard 7 with SP1

***Note**

- ✓ Windows Embedded Standard 7은 MSI 설치가 필요합니다.
- ✓ Microsoft HALs는 아래 목록 중 하나가 Windows 장치 관리자의 컴퓨터 항목에 설치되어야 합니다.
  - ACPI Uniprocessor PC
  - ACPI Multiprocessor
  - ACPI x86-based PC
- ✓ 2018년 1월 이후 KB 업데이트가 설치된 Windows 7에서는 RTX2016의 업데이트 3이 설치되어 있고, KB3033929 Windows 업데이트도 설치되어 있어야 합니다 관련 자료는 아래의 "Intervalzero RTX 2016 Downloads" 링크를 참조하시길 바랍니다.
- ✓ Windows 10 빌드 버전에 따른 RTX 호환성 정보는 아래의 "Intervalzero RTX64 Support Windows 10 Updates" 링크를 참조하시길 바랍니다.

	Intervalzero RTX Reference Link
RTX Update	Intervalzero RTX 2016 Downloads
	Intervalzero RTX64 Support for Windows 10 Updates

1.3.1.3. NIC (Network Interface Card)

WMX3 는 RtlGB 호환 NIC 을 추천합니다. 각 드라이버 별로 지원하는 NIC 은 아래 리스트를 참조하십시오.
"Tested" 표시되지 않은 NIC 은 당사에서 확인되지 않았지만, 이론적으로는 칩셋의 사양에 따라 작동될 것입니다.

***Note**

- ✓ WMX3 Windows 전용 버전은 정상적으로 인식되는 모든 NIC을 사용할 수 있습니다.

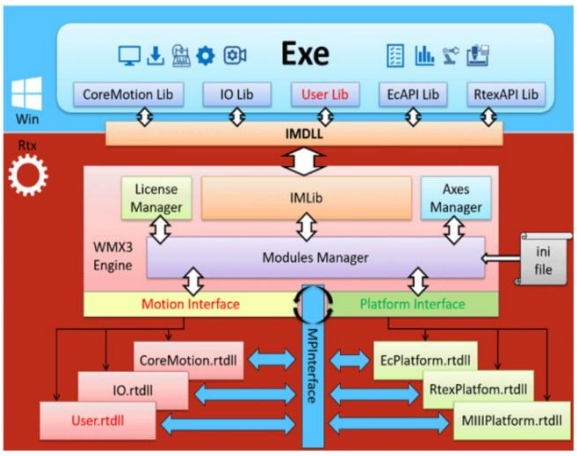
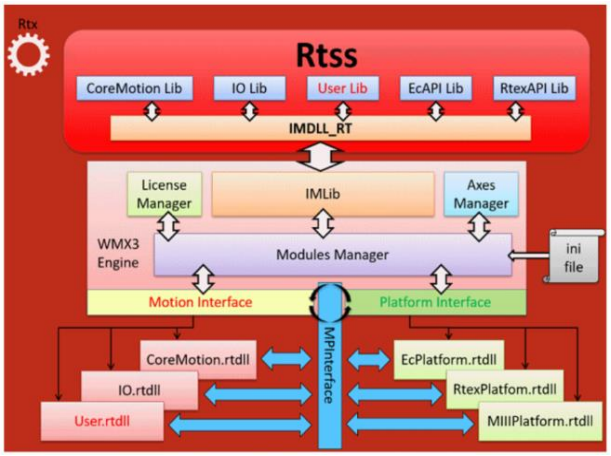
	Vendor ID	Device ID	Device Name	Tested	Remarks
RtlGB	0x8086	0x1533	Intel I210 T1 Copper-only Ethernet Controller	Done	
	0x8086	0x1521	Intel I350 Quad/Dual PCIe Copper Ethernet Controller	Done	No MSI-X
	0x8086	0x1523	Intel I350 Quad/Dual PCIe 1000BASE-KX/BX Ethernet Controller		No MSI-X
	0x8086	0x1539	Intel I211-AT Ethernet Controller	Done	
	0x8086	0x157B	Intel I210 Flash-less Copper-only Ethernet Controller		
RtlPCH	0x8086	0x153A	Intel I217LM PHY 1000BASE with C220 Ethernet Controller	Done	
	0x8086	0x153B	Intel I217V PHY 1000BASE with C220 Ethernet Controller	Done	
	0x8086	0x15A1	Intel I218V PHY 1000BASE with Ethernet Controller	Done	
	0x8086	0x1526	Intel Gigabit ET2 Quad Port Server Adapter	Done	
	0x8086	0x10C9	Intel 82576 Gigabit ET/ET2/EF Server Adapter	Done	
	0x8086	0x150E	Intel 82580 Quad Port 10/100/1000 Mb/s Ethernet Controller	Done	
	0x8086	0x15B8	Intel PCH SPT I219 V2 Ethernet Controller	Done	
	0x8086	0x1502	Intel 82579 Ethernet Controller Integrated with the Intel C202, C216, C220 PCH	Done	Not C200

1.4. 아키텍처

이 장은 WMX3 아키텍처에 대한 자세한 내용을 기술합니다.

1.4.1. 아키텍처 개요

WMX3 아키텍처는 유연한 구조로 이루어져 있기 때문에 새로운 기능이나 기존 기능을 확장하는데 매우 용이합니다. 모든 기능은 rtdll 이라는 라이브러리에 구현 되어있으며 이를 불러와 해당 함수를 사용할 수 있습니다. 또한 사용하지 않는 모듈의 경우 비활성화하여 메모리 공간을 더 효율적으로 사용할 수 있습니다.

	윈도우 어플리케이션을 통해 RTX 를 구동할 경우의 구조	*RTX 에서 단독으로 구동할 경우의 구조*
Architecture		

WMX3 구조상의 이점

- 모듈간 데이터 교환이 매우 빠릅니다.
- 다수의 Platform Modules 이 사용될 경우에 hybrid 형태의 통신 프로토콜 구현이 가능합니다.
- 유저가 만든 어플리케이션을 바꿀 필요 없이 다른 Platform Module 로 바꾸는 것이 가능합니다.
- 새로운 기능 개발 또는 문제점들을 수정 시 전체 시스템을 업데이트 할 필요가 없이 해당 .dll 파일만 수정 가능합니다.
- User.rtdll 을 통해 유저가 원하는 기능을 쉽게 구현할 수 있으며, WMX3 에서 미리 정의된 Interface 를 제공하고 있기 때문에 사용자는 필요한 Interface 의 내부를 올바른 시기에 채워 넣기만 하면 됩니다. 이에 사용자가 굳이 시스템 내부에 대해 깊게 고민할 필요 없이 정말 중요한 코드에 집중할 수 있습니다.

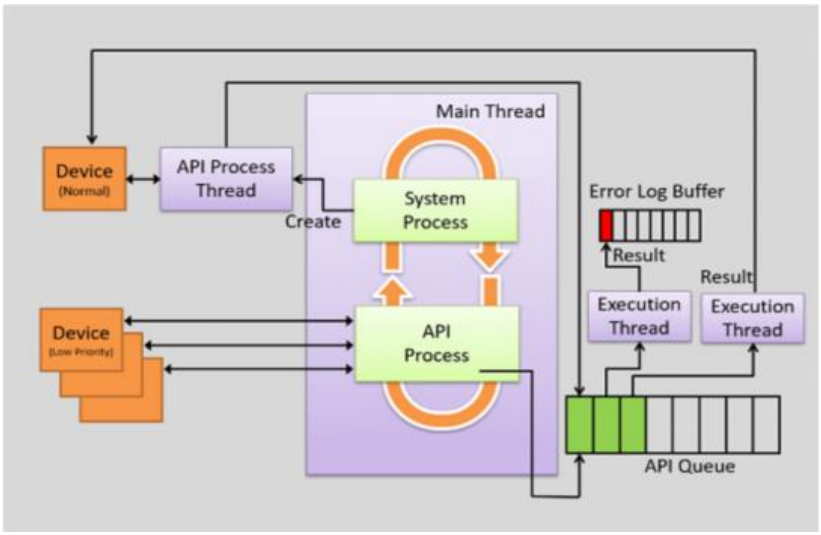
1.4.2. 아키텍처 Device

사용자의 어플리케이션은 WMX3 의 엔진과 해당 모듈들과 "Devices"를 통해 통신합니다. API 기능 중 CreateDevice 는 새로운 Device 를 생성할 때 필요로 합니다. 그리고 이렇게 생성된 Device 를 통해, API 로부터 전달된 커맨드가 WMX3 엔진으로 전송되고 해당 프로세스에 대한 결과 값이 반환됩니다. Device 는 아래의 두가지 타입으로 생성할 수 있습니다.

	Type	*Description*
Device	Normal	Normal Device 의 경우 명령을 처리하고 응답하는 속도가 매우 빠릅니다. 각 Device 마다 전용 API 프로세스 쓰레드가 할당됩니다.
	Low Priority	Low Priority Device 의 경우 System Process 가 끝난 후에 메인 쓰레드에서 하나씩 처리합니다. 만약에 메인 쓰레드에서 이미 명령을 처리하고 있는 상태라면 다른 Device 로부터온 명령은 현재명령이 다 처리될 때까지 기다려야 합니다. 따라서 Low Priority Devices 는 빠른 처리와 응답 속도를 요구하지 않는 상황에서 사용하길 권장합니다.

Normal, Low Priority 의 Devices 가 API 쓰레드와 주고받는 데이터 흐름도

Architecture



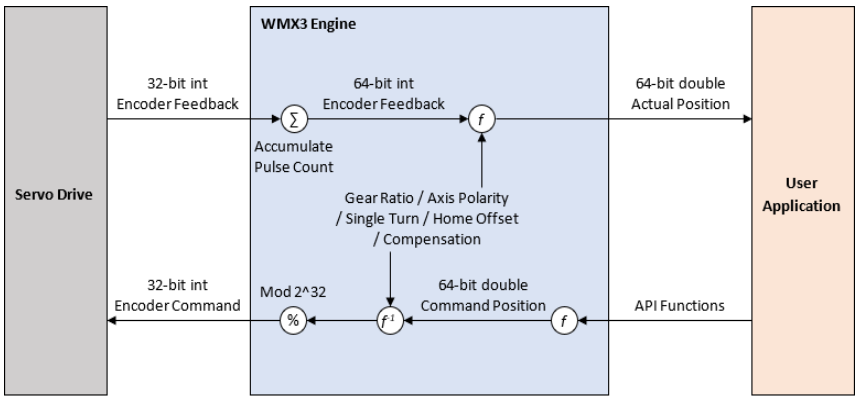
커맨드는 타입에 따라 API Process Thread 에 의해 즉시 처리되거나 또는 API Queue 에 추가되어 Execution Thread 에 의해 실행이 되기를 기다립니다. 후자의 경우, Execution Thread 가 모드 커맨드를 처리하고 결과 값을 반환하기 전까지 Device 로부터 다음 실행을 계속해서 이어 나가지 못하게 막습니다.

이렇게 하는 이유는 Device 에서 실행한 이전 커맨드가 온전히 다 실행되지 않은 상황에서 Execution Thread 를 벗어나 다음 커맨드를 실행할 수 있기 때문입니다. 만약 이런 상황이 발생할 경우에는 Error Log Buffer 에 해당 커맨드에 대한 에러 코드가 기록되며 추후에 받아볼 수 있습니다.

1.4.3. 아키텍처 Axis Position

WMX 시스템에서 축 위치 데이터의 흐름

Architecture



Encoder Feedback 이란 Servo Drive 로부터 전달된 위치 값입니다. 이 위치 값은 32-bit 로 이루어진 정수 데이터로서, 32-bit 값을 넘어서는 멀티 턴 또는 홈 위치를 고려하지 않습니다. 위치 값의 기본 단위는 펄스입니다. 위치 값은 Encoder 의 커맨드 상태와 같은 방식으로 Getstatus 기능을 통해 확인할 수 있습니다.

Encoder Feedback 에서는 매 사이클마다 변화하는 위치 값이 64-bit 정수 Encoder Feedback Position 에 연산 되어 더해집니다. 여기서 사용되는 위치 값 역시 Pulse 를 기본 단위로 사용합니다. 이는 Multi Turn 축이 가질 수 있는 범위를 나타내는데 사용됩니다. 만약 어떤 한 축이 64-bit 를 넘어서는 값을 가질 경우, 피드백 위치는 더이상 업데이트 하지 않습니다. (표현 가능한 위치 범위를 넘어서기 때문입니다) Single Turn 축의 경우에는 한계가 없기 때문에 어떠한 방향으로도 (Feedback Position 이 멈출 필요 없이) 무한대로 움직일 수 있습니다. 이러한 64-bit Encoder Feedback 은 내부 몇몇의 Parameter 를 통해 Actual Position 으로 재 표현됩니다. 여기서의 단위는 사용자에 의해 정의되며, 기어비에 Pulse 를 곱한 값입니다.

Actual Position 로 전환하는데 사용하는 Parameters, Functions.

Gear Ratio Numerator & Denominator	모터의 실제 분해능과 사용자 정의 단위로서 Actual Position 을 결정짓는 인수입니다.
Single Turn Mode & Single Turn Encoder Count	특정 축을 Single Turn Rotary 으로 설정하여 한바퀴 돌 때마다 위치 값을 0 으로 표시합니다.
Axis Polarity	모터의 축 방향을 제어할 수 있습니다.
Home Offset	축의 홈 위치를 설정할 때 적용됩니다. Home Offset 은 StartHome 을 통해 홈 위치로 가거나, SetCommandPos 또는 SetFeedbackPos 가 실행되었을 때 업데이트 됩니다.
Compensation	Pitch Error Compensation, 2D Pitch Error Compensation, Backlash Compensation 의 총합을 의미합니다. 이러한 Compensations 은 SetPitchErrorCompensation, Set2DPitchCompensation, 그리고 SetBacklashCompensation 으로 설정할 수 있습니다.

위의 Parameters 혹은 Functions 에 의한 변화가 있을 때, 이는 즉시 64-bit 정수형 Encoder Feedback 에 반영되어 Actual Position 에 영향을 줍니다. Command Position 은 Actual Position 과 같은 좌표에 속해 있고. 이 위치 값은 사용자 단위를 사용합니다. Command Position 은 해당 축에 모션 기능을 실행할 때 업데이트되며 Pos Cmd Status 를 통해 값을 반환합니다. Actual Position 을 나타내는 64-bit 정수 Encoder Feedback 에서 사용되는 Parameters 은 사용자의 Command Position 을 Modulo 2^32 함수를 통해 32-bit 정수 Encoder Command 로 변환하는데 사용됩니다. Parameters 의 값을 변경할 경우, Command Position 에도 영향을 주게 되지만, Encoder Command 는 바뀌지 않습니다. 즉, Parameters 를 변경하면 Command Position 의 값 또한 바뀌지만, 해당 축이 움직이지는 않습니다. Encoder Command Position 은 펄스 단위입니다. 이 값은 Encoder Command Status 를 통해 반환 받을 수 있습니다.

1.4.4. 아키텍처 Interrupt

인터럽트는 Servo Network 와 통신하기위해 주기적으로 실행됩니다. 인터럽트의 역할은 주기적으로 실행되어 Feedback 값을 읽고 Command 를 Servo Network 로 송신하는데 있습니다. 인터럽트는 하나의 Master 로부터 인터럽트의 최대 개수만큼 정의되어 Servo Network 를 관리할 수 있습니다. 여러 개의 인터럽트를 사용하기 위해서는, Module.ini 파일에 저장된 NumOfMaster 와 NumOfInterrupt Parameters 가 적절하게 구성되어야 합니다. 이에 관련해서 Module Configuration (Module.ini)을 참고하세요.

	Interrupt 관련 Axis, Device
Interrupt of an Axis	모든 축은 하나의 인터럽트를 할당 받습니다. 축에 대한 Command 와 Feedback 은 매번 인터럽트 루틴에서 업데이트 됩니다.
Interrupt of a Device	CreateDevice 를 통해 생성된 모든 Device 는 인터럽트와 연관이 있습니다. 하지만 영향을 받는 기능의 수는 매우 적습니다. 이렇게 영향을 받는 기능들을 제외하면, 인터럽트는 Device 의 구동에는 별다른 영향을 미치지 않습니다.
****Note***	✓ 어떤 Device에서든 Device와 축에 관련된 인터럽트가 서로 다르더라도 대부분의 Motion Command를 실행할 수 있습니다. ✓ Device에 연결된 인터럽트 설정을 바꾸려면 SetInterruptId API를 사용하십시오.

Interrupt 관련 함수 일람

영역별로 각 모듈에 대한 기능 중 Device 와 명령을 내리는 축에 영향을 미치는 인터럽트와 관련된 함수 일람입니다.

	축 관련 인터럽트에 의해 영향을 받는 함수
AdvancedMotion	StartECAM SetConstantLinearVel StartCSplinePos StartCSplineMov StartCBSplinePos StartCBSplineMov StartPVT (PVTIntplCommand *pPVTCommand) StartPathIntplPos StartPathIntplMov StartPathIntpl3DPos StartPathIntpl3DMov SetPathIntplWithRotationConfiguration SetPathIntplLookaheadConfiguration StartPathIntplLookahead StartCoordinatedPos (CoordinatedPosCommand *pPosCommand) StartCoordinatedPos (unsigned int numCommands, CoordinatedPosCommand *pPosCommand) StartCoordinatedPos (CoordinatedJerkRatioPosCommand *pPosCommand) SimulatePosAtTime SimulateTimeAtPos SimulateTimeAtDist StartCBSplinePos
****Note***	✓ Sync와 Interpolation 함수들은 모든 축이 동일한 인터럽트가 아닌 다른 곳에서 실행될 경우 InterruptMismatch Error를 반환합니다. ✓ *pPosCommand의 경우 두 축이 동일한 좌표의 Position Command를 가지지 않는 이상 서로 다른 인터럽트를 가집니다.

Device 관련 인터럽트에 영향을 받는 함수

ApiBuffer

Evaluation of wait conditions
Detection of watch errors
Execution of watch error routines

***Note** ✓ 각 ApiBuffer Channel은 인터럽트와 연관되어 있습니다. 실행 함수가 호출될 때, 해당 Channel은 함수를 호출하는 Device의 인터럽트와 연결됩니다. 하기 프로세스는 인터럽트의 주기적인 루틴 하에 처리됩니다.

축 관련 인터럽트에 의해 영향을 받는 함수

Compensation

Set2DPitchErrorCompensation

***Note** ✓ Compensation 함수는 모든 축이 동일한 인터럽트가 아닌 다른 곳에서 실행될 경우 InterruptMismatch Error를 반환합니다.

축 관련 인터럽트에 의해 영향을 받는 함수

CoreMotion

SetSyncMasterSlave
SetSyncCombine
SetSyncGearRatio (int masterAxis, int slaveAxis, double gearRatio, Profile *pProfile)
StartLinearIntplPos
StartLinearIntplMov
StartCircularIntplPos
StartCircularIntplMov
StartHelicalIntplPos
StartHelicalIntplMov
SimulateLinearIntplPos

- SimulatePosAtTime (SimulateLinearIntplCommand *pSimulateLinearIntplCommand, double timeMilliseconds, double *pPosArray, double *pMoveDistance, double *pRemainDistance, double *pTotalDistance)
- SimulateTimeAtDist (SimulateLinearIntplCommand *pSimulateLinearIntplCommand, double specificDistance, double *pMoveTimeMilliseconds, double *pRemainTimeMilliseconds, double *pTotalTimeMilliseconds)

***Note** ✓ Sync와 Interpolation함수는 모든 축이 동일한 인터럽트가 아닌 다른 곳에서 실행될 경우 InterruptMismatch Error를 반환합니다.

Device 관련 인터럽트에 영향을 받는 함수

CoreMotion

Motion :: Wait (int axis)
Motion :: Wait (AxisSelection *pAxisSelection)
Motion :: Wait (WaitCondition *pWaitCondition)
Motion :: Wait (int axis, unsigned int waitTimeMilliseconds)
Motion :: Wait (AxisSelection *pAxisSelection, unsigned int waitTimeMilliseconds)
Motion :: Wait (WaitCondition *pWaitCondition, unsigned int waitTimeMilliseconds)

***Note** ✓ Wait 함수들은 Device 관련 인터럽트와 동기화되어 호출되며, Cyclic Handler의 루틴에 있는 Wait 조건이 충족된 직후 타이밍에 맞춰 반환합니다.

인터럽트에 영향을 받는 다른 함수들

CoreMotion	Getstatus	주기적으로 얻은 상태 값들은 모든 인터럽트의 마지막 루틴에 업데이트 됩니다.
	cycleTimeMilliseconds	상태 값이 각 인터럽트에 반환됩니다.
	cycleCounter	상태 값이 각 인터럽트에 반환됩니다.
	Flight recorder	각 인터럽트마다 개별적으로 동작합니다. Flight Recorder 파일 뒤에 "_ipt[n]" (n = 인터럽트 아이디)가 붙습니다.

		Device 관련 인터럽트에 영향을 받는 함수
Event	Events	축 상태와 연관된 조건에 맞게 해당 축이 할당 받은 인터럽트의 주기적인 루틴 하에 처리됩니다. 다른 Events 는 0 번째 인터럽트에 의해 처리됩니다.
	Touch Probe	Z-pulse 관련 함수는 Position Data 를 latch 에 등록하기 위해, 해당 축의 인터럽트의 주기적인 루틴에 동기화 되어 있습니다. 서로 다른 인터럽트가 할당된 여러 축 간에 Touch Probe Channel 을 변경하려면, 우선 Touch Probe Channel 가 비활성화되어야 합니다.
	Position synchronous output process	Position 값이 참조 되어있는 축의 인터럽트의 주기적인 루틴과 동기화 되어있습니다.
	Planned velocity override process	오버라이드된 velocity 를 지닌 축의 인터럽트의 주기적인 루틴과 동기화 되어있습니다.
		Device 관련 인터럽트에 영향을 받는 함수
IO	SetInitialOutByte SetInitialOutBytes GetInitialOutByteInterruptId GetInitialOutBytesInterruptId	
	****Note***	✓ 모든 출력 byte는 통신이 시작되었을 때 초기 출력 상태로 설정되도록 구성되어있습니다. 출력 byte의 상태가 초기 출력 상태로 설정될 때 인터럽트와 연관지어집니다. 그리고 인터럽트가 Servo Network와 통신을 시작할 때 비로소 초기 출력 상태가 적용됩니다. ✓ 초기 출력을 설정하는 SetInitialOutByte, SetInitialOutBytes 함수들은 출력 byte와 이런 함수들을 호출하는 Device에 연관된 인터럽트와 관련이 있습니다. ✓ GetInitialOutByteInterruptId, GetInitialOutBytesInterruptId 함수는 출력 byte와 관련된 인터럽트의 ID를 받아옵니다.
		축 관련 인터럽트에 의해 영향을 받는 함수
Log	SetLog SetMemoryLog	
	****Note***	✓ 위 함수들은 지정된 모든 축이 동일한 인터럽트에 속해 있지 않을 경우 InterruptMismatch 에러를 반환합니다.
		Device 관련 인터럽트에 영향을 받는 함수
Log	GetLogStatus GetMemoryLogStatus	
	****Note***	✓ Log Channel이 아직 구성되지 않은 경우, 위 함수들은 Cyclic Time을 받아오기 위해 자신을 호출하는 Device에 할당된 인터럽트를 참조합니다.

1.4.5. 아키텍처 Blocking

WMX3 라이브러리에 있는 대부분의 함수들은 Non-Blocking 입니다. Non-Blocking 함수들은 실행을 마치면 Micro 시간 내에 자신을 호출한 Thread 로 반환됩니다. 하지만, 몇몇의 함수들 또는 조건들은 함수로 하여금 하나 이상의 통신 주기 동안 멈추게 합니다. 이러한 딜레이가 왜 생기는지 그리고 이에 대한 해결 방안은 무엇인지에 대해 이번 페이지에서 논의하고 있습니다.

인터럽트 진행 도중 함수가 호출되는 경우

WMX3 엔진은 매 주기마다 인터럽트로 하여금 Servo Network 와 통신하도록 진행됩니다. 이때 만약 WMX3 라이브러리 함수가 호출될 경우, 혹은 함수 실행이 다 완료되지 않았을 때 인터럽트가 진행될 경우, 호출된 함수는 인터럽트가 마저 다 끝날 때까지 딜레이 됩니다. 인터럽트 진행에 소요되는 시간은 CPU 의 속도, Servo Network 에 붙은 축의 개수, 그리고 지정된 축에서 실행 중인 함수들의 영향을 받습니다. 특정 함수들의 경우 엔진과 통신할 필요가 없으며 이런 함수들은 엔진이 인터럽트를 처리하고 있어도 딜레이 없이 즉각적인 실행이 가능합니다. 딜레이를 피하는 다른 방법은 affinityMask 라는 전달인자를 받는 함수 (WMX3 엔진을 시작하는 함수로, CreateDevice, StartEngine, RestartEngine 이 있습니다)를 통해 RTX Subsystem 에 더 많은 코어를 할당하는 것입니다. 이로써 RTX Subsystem 에 할당된 첫번째 코어는 인터럽트를 진행하고, 두번째 코어는 딜레이 없이 함수 호출이 가능해집니다. 이는 둘 이상의 추가 코어를 사용하는 RTX License 를 요구합니다.

	WMX3 엔진과 통신이 필요 없는 함수
Blocking	ErrorToString (of every module)
	GetLibVersion (of every module)
	WMX3Api :: GetIMDllVersion
	CoreMotion :: GetStatus
	Io :: SetOutBitEx
	Io :: SetOutByteEx
	Io :: SetOutBytesEx
	Io :: SetOutBitsEx
	Io :: SetOutAnalogDataCharEx
	Io :: SetOutAnalogDataUCharEx
	Io :: SetOutAnalogDataShortEx
	Io :: SetOutAnalogDataUShortEx
	Io :: SetOutAnalogDataIntEx
	Io :: SetOutAnalogDataUIntEx
	Io :: GetInBitEx
	Io :: GetInByteEx
	Io :: GetInBytesEx
	Io :: GetInAnalogDataCharEx
	Io :: GetInAnalogDataUCharEx
	Io :: GetInAnalogDataShortEx
	Io :: GetInAnalogDataUShortEx
	Io :: GetInAnalogDataIntEx
	Io :: GetInAnalogDataUIntEx
	Io :: GetOutBitEx
	Io :: GetOutByteEx
	Io :: GetOutBytesEx
	Io :: GetOutAnalogDataCharEx
	Io :: GetOutAnalogDataUCharEx
	Io :: GetOutAnalogDataShortEx
	Io :: GetOutAnalogDataUShortEx
	Io :: GetOutAnalogDataIntEx
	Io :: GetOutAnalogDataUIntEx

"Wait Until Motion Start" Parameter

Wait Until Motion Start Parameter 는 Default 값으로 True 로 설정되어 있으나 SetParam 또는 이 Parameter 값을 바꾸는 함수를 통해 Default 값을 바꿀 수 있습니다.

	Value	Description
Wait Until Motion Start Parameter	True	Wait Until Motion Start Parameter 가 True 로 설정되어 있는 경우, 대부분의 모션 함수는 다음 인터럽트가 끝나기 전까지 딜레이 됩니다. 이는 사용자로 하여금 같은 쓰레드 안에서 Override 모션 함수를 실행하기 전에 이미 모션이 동작 중에 있는지 체크할 필요가 없어집니다.
	False	Wait Until Motion Start Parameter 가 False 로 설정되어 있는 경우, 또는 Override 모션 함수가 다른 쓰레드에서 실행될 경우, Command Ready 상태를 반드시 체크하여야 합니다. 만약 Override 모션 함수가 첫번째 인터럽트 이전에 모션이 시작되고나서 호출될 경우 StartingPreviousCommand Error 를 반환합니다.
	***Note**	✓ Cylce Time Milliseconds가 1ms일때, StartPos를 호출하고 나서 1ms가 지나기 전에 또 다시 StartPos를 실행할 경우 이와 같은 에러를 반환합니다.

	Wait Until Motion Start Parameter 에 영향을 받는 함수
Wait Until Motion Start Parameter	Motion :: StartPos
	Motion :: StartMov
	Motion :: StartLinearIntplPos
	Motion :: StartLinearIntplMov
	Motion :: StartCircularIntplPos
	Motion :: StartCircularIntplMov
	Motion :: StartHelicalIntplPos
	Motion :: StartHelicalIntplMov
	Motion :: StartJog
	Motion :: StartPosToJog
	Motion :: StartMovToJog

Motion :: Stop
Motion :: ExecQuickStop
Motion :: ExecTimedStop
Motion :: Pause
Motion :: Resume
Motion :: OverridePos
Motion :: OverrideMov
Motion :: OverrideVel
Motion :: OverrideAcc
Motion :: OverrideDec
Motion :: OverrideJerkAcc
Motion :: OverrideJerkDec
Motion :: OverrideProfile
Motion :: StopJogAtPos
Velocity :: StartVel
Velocity :: Stop
Velocity :: ExecQuickStop
Velocity :: StartPosToVel
Torque :: StartTrq
Torque :: StartRampTimeTrq
Torque :: StartRampRateTrq
Torque :: StartPosToTrq
Torque :: StartVelToTrq
AdvMotion :: StartCSplinePos
AdvMotion :: StartCSplineMov
AdvMotion :: StartCBSplinePos
AdvMotion :: StartCBSplineMov
AdvMotion :: StartPVT
AdvMotion :: StartPT
AdvMotion :: StartVT
AdvMotion :: StartAT
AdvMotion :: StartPathIntplPos
AdvMotion :: StartPathIntplMov
AdvMotion :: StartPathIntpl3DPos
AdvMotion :: StartPathIntpl3DMov
AdvMotion :: StartPathIntplWithRotation
AdvMotion :: StartPathIntplLookahead
AdvMotion :: StartCoordinatedPos
AdvMotion :: StartTwoLinkLinearPos
AdvMotion :: StartTwoLinkLinearMov
AdvMotion :: StartTwoLinkRotaryPos
AdvMotion :: StartTwoLinkRotaryMov
AdvMotion :: StartTwoLinkUntetheredLinearPos
AdvMotion :: StartTwoLinkUntetheredLinearMov
AdvMotion :: StartTwoLinkUntetheredRotaryPos
AdvMotion :: StartTwoLinkUntetheredRotaryMov

Wait Functions

CoreMotion 모듈에 있는 Wait 함수들은 특정 조건이 만족되기 전까지는 반환하지 않고 기다립니다. Wait 함수들을 실행하는 Thread 은 조건이 충족되기 전까지 잠시동안 멈춥니다.

Path Interpolation with Look Ahead

Path Interpolation with Look Ahead 의 경우 처리하는데 많은 시간을 요구하며, 특히 이동할 Points 수 가 많을 수록 더 오래 걸립니다. AddPathIntplWithRotationCommand 함수는 Points 의 양에 따라 짧게는 몇 백 ms 에서 몇 천 ms 동안 딜레이에 빠질 수 있습니다.

RemoveEvent and ClearAllEvent

RemoveEvent 함수와 ClearAllEvent 함수는 다음 인터럽트가 다 끝날 때까지 기다립니다. 인터럽트가 진행되는 도중에는 특정 이벤트가 제거됩니다. 통신이 멈추고 이벤트가 처리되지 않고 있을 때, 이 두 함수는 딜레이 되지 않고 즉시 특정 이벤트를 제거해줍니다.

1.5. EtherCAT

1.5.1. EtherCAT 개요

1.5.2. EtherCAT 사양

1.5.3. EtherCAT 기능 및 특징

1.6. 모션 기능

이 장은 WMX3 에서 지원하는 모션 관련 기능에 대하여 자세히 설명합니다. WMX3 Motion API 사용에 앞서 모션 기능에 대한 내용을 숙지하시어 올바르게 사용하시기 바랍니다.

1.6.1. 기본 모션

WMX3 의 기본적인 모션에 대하여 설명합니다. 해당 장에서 기술된 모션 기능은 WMX3 Standard 라이선스 이용 시 사용이 가능합니다.

1.6.1.1. 모션 프로파일 개요

WMX3 에서는 여러 가지 유형의 모션 프로파일을 지원합니다. 모션 프로파일은 축이 목표 위치 또는 조그, 속도 지령으로 목표 속도로 이동할 때 축의 속도, 가속도 및 저크가 어떻게 변화하는지를 결정합니다.

1.6.1.1.1. 프로파일 파라미터

모션 프로파일과 관련된 파라미터는 Profile 클래스 안에서 지정할 수 있으며 각 프로파일 타입은 모든 파라미터를 사용하지는 않습니다. 모션 프로파일의 형태는 목표 위치, 속도, 설정한 파라미터에 의해 결정됩니다. 각각의 프로파일 파라미터는 아래에 내용을 참조하세요.

Profile Type

프로파일 타입은 모션 프로파일의 형태를 전체적인 모양을 결정합니다. 각각의 프로파일 타입은 모션 프로파일에서 상세히 설명합니다.

Variable Name	type
Type	ProfileType
Default Value	Trapezoidal

Velocity

프로파일의 속도는 모션이 이동중 각 경로의 속도를 결정합니다. 이 수치는 축 모션의 방향에 상관없이 양수의 값을 갖습니다.

Variable Name	velocity
Type	double
Unit	user unit / second
Signed / Unsigned	unsigned
Minimum Value	1e-6 = 0.000001
Maximum Value	2^38-1 = 274877906943

Acceleration

프로파일의 가속도는 모션경로의 가속도를 결정해 줍니다. 이 수치는 축 모션의 방향에 상관없이 양수의 값을 갖게 됩니다. 프로파일 타입에 따라서, 이 파라미터는 평균 가속도나 최대 가속도로를 의미할 수 있습니다.

Variable Name	acc
Type	double
Unit	user unit / second^2
Signed / Unsigned	unsigned
Minimum Value	1e-6 = 0.000001
Maximum Value	2^38-1 = 274877906943

Deceleration

프로파일 감속도는 모션 수행 중의 감속도를 결정합니다. 이 수치는 축 모션의 방향에 상관없이 양수의 값을 갖게 됩니다. 프로파일 타입에 따라서, 이 파라미터는 평균 가속도나 최대 감속도로를 의미할 수 있습니다. 감속도 파라미터는 아래 부분에 적용이 되어집니다.

- 모션의 마지막 이동경로의 감속도
- 오버라이딩 위치에서 현재의 속도보다 더 느리게 프로파일 속도가 설정되어 있을 때의 축 속도를 감속시키는 감속도
- 오버라이드 위치에 모션을 시작 시 이동 방향과 반대 방향으로 목표 위치가 있을 경우 축을 정지시키기 위한 감속도 (Prohibit Overtravel 파라미터의 Disabled 로 설정되어야 함)

Variable Name	dec
Type	double
Unit	user unit / second^2

Signed / Unsigned	unsigned
Minimum Value	1e-6 = 0.000001
Maximum Value	2^38-1 = 274877906943

Acceleration Jerk

가속하는 동안 가속 저크값을 결정합니다. 이 수치는 축 모션의 방향에 상관없이 양수의 값을 갖게 됩니다.

Variable Name	jerkAcc
Type	double
Unit	user unit / second^3
Signed / Unsigned	unsigned
Minimum Value	1e-6 = 0.000001
Maximum Value	2^38-1 = 274877906943

Deceleration Jerk

감속하는 동안의 감속저크의 값을 결정합니다. 이 수치는 축 모션의 방향에 상관없이 양수의 값을 갖게 됩니다.

Variable Name	jerkDec
Type	double
Unit	user unit / second^3
Signed / Unsigned	unsigned
Minimum Value	1e-6 = 0.000001
Maximum Value	2^38-1 = 274877906943

Acceleration Jerk Ratio

프로파일의 가속 저크 비율을 결정하면 단위는 시간 비율로 결정됩니다. 이 비율은 0 과 1 사이에서 결정됩니다. (예를 들면 0.25, 0.5 와 같은 수치) 만약, 저크 비율이 0 일때 가속하는 구간은 TrapeZodial 과 같게 됩니다. 저크 비율이 1 인 경우 가속 구간 전체에서 SCurve 와 같은 모양을 갖게 됩니다.

Variable Name	jerkAccRatio
Type	double
Minimum Value	0
Maximum Value	1

Deceleration Jerk Ratio

프로파일의 가속 저크 비율을 결정하면 단위는 시간 비율로 결정됩니다. 이 비율은 0 과 1 사이에서 결정됩니다. (예를 들면 0.25, 0.5 와 같은 수치) 만약, 저크 비율이 0 일때 감속하는 구간은 TrapeZodial 과 갖게 됩니다. 저크 비율이 1 일때는 감속 구간 전체에서 SCurve 와 같은 모양을 갖게 됩니다.

Variable Name	jerkDecRatio
Type	double
Minimum Value	0
Maximum Value	1

Acceleration Time

프로파일의 가속 시간은 전체 가속 시간을 결정합니다.

Variable Name	accTimeMilliseconds
Type	double
Unit	millisecond
Minimum Value	1
Maximum Value	2^31-1 = 2147483647

Acceleration Time

프로파일의 감속 시간은 전체 감속 시간을 결정합니다.

Variable Name	decTimeMilliseconds

Type	double
Unit	millisecond
Minimum Value	1
Maximum Value	2^31-1 = 2147483647

Starting Velocity

프로파일의 시작 속도는 모션을 시작하는 초기속도를 결정합니다. 이 수치는 축의 방향에 관계없이 양수의 값을 갖게 됩니다. 이 수치는 0 으로 설정되어질 수 있습니다. 이 경우 축의 현재 속도로 설정이 됩니다. Idle 상태에서 시작한 모션 명령은 초기 속도가 0 이 될 것입니다. 오버라이드 모션 명령에서는 초기 속도가 축의 현재 속도가 될 것입니다.

Variable Name	startingVelocity
Type	double
Unit	user unit / second
Signed / Unsigned	unsigned
Minimum Value	1e-6 = 0.000001
Maximum Value	2^31e-6 = 0.000001
Special Value	0

End Velocity

프로파일의 End 속도는 모션을 마지막속도를 결정합니다. 이 수치는 축의 방향에 관계없이 양수의 값을 갖게 됩니다. 이 수치는 0 으로 설정되어질 수 있습니다. 이 경우 축의 끝부분은 부드럽게 멈출 것입니다. 하지만 0 이 아닌 값으로 설정이 되었 있으면 축은 갑자기 End 속도에서 0 으로 감속할 것입니다. 이 수치는 프로파일 속도보다 작아야 합니다. 만약 프로파일 속도보다 크다면 프로파일 속도와 같도록 설정이 되어집니다. 만약, End 속도까지 가감속도 하기 전에 목표 위치까지 도달한다면, 모션은 가능한한 End 속도에 가까운 수치로 끝나게 됩니다.

Variable Name	endVelocity
Type	double
Unit	user unit / second
Signed / Unsigned	unsigned
Minimum Value	1e-6 = 0.000001
Maximum Value	2^31e-6 = 0.000001
Special Value	0

Second Velocity

프로파일의 Second 속도는 특정 프로파일 타입을 만들기 위한 Second 속도를 결정합니다. 이 수치는 축의 방향에 관계없이 양수의 값을 갖게 됩니다. 이 수치를 적용하는 프로파일 타입은 일정한 이동 속도로 움직이는 대신에 정해진 가속도에 의해 Second 속도로 가속하게 되어집니다. 이 수치는 프로파일 속도보다 커야 합니다. 만약 프로파일 수치보다 작다면, 그것은 프로파일 속도로 맞춰집니다.

Variable Name	secondVelocity
Type	double
Unit	user unit / second
Signed / Unsigned	unsigned
Minimum Value	1e-6 = 0.000001
Maximum Value	2^38-1 = 274877906943

Moving Average Time

프로파일의 평균 동작시간은 TrapezoidalMAT 프로파일 타입에 적용된 평균 필터 동작의 일정 시간을 결정합니다. 만약, 프로파일이 높은 시작, End 속도에 의해 설정이 되어진 경우 프로파일 형성할 수 없을 경우 movingAverageTimeMilliseconds 에 의해 설정되게 됩니다.

Variable Name	movingAverageTimeMilliseconds
Type	double
Unit	millisecond
Minimum Value	0
Maximum Value	120000 (120 seconds)

1.6.1.1.2. 최대 이동거리

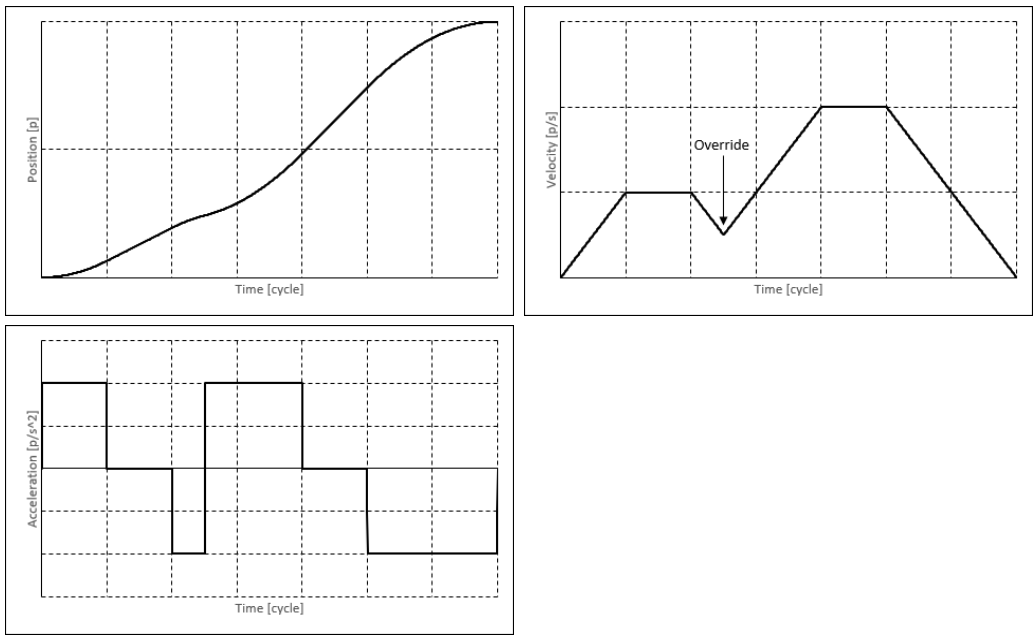
최대 이동거리는 현재 위치부터 최고 2^38-1 사용자 유닛으로 지정할 수 있습니다. 더 먼 거리를 이동하기 위해서는 Multiple Position 수행해야만 합니다. 축이 정지함이 없이 이동하기 위해서는 오버라이딩 위치 명령을 사용합니다. 최대 이동 경로는 보간 부분의 기본 보간 명령에 적용이 되어집니다. 보간 거리는 이 수치보다 낮아야 합니다. 만약 더 큰 수치 값이 위치 명령에 적용되었다면 그 축은 이동하지 않을 것입니다.

1.6.1.1.3. 오버라이드 프로파일

많은 모션 명령(위치, 조그, 보간 명령)들은 같은 타입의 모션명령에 의해 오버라이드 될 수 있습니다. 오버라이드 동안에 축의 위치나 속도는 (프로파일이나 가속도) 정지함이 없이 새 명령을 수행됩니다. 만약, 시작 오버라이드 시작 속도가 0 이 아니다면, 축의 속도는 오버라이드 시간에 속도를 유지하는 대신에 특정한 시작 속도로 오버라이드 될 것입니다.

오버라이드 모션 명령의 위치, 속도, 가속도

Override Profiles

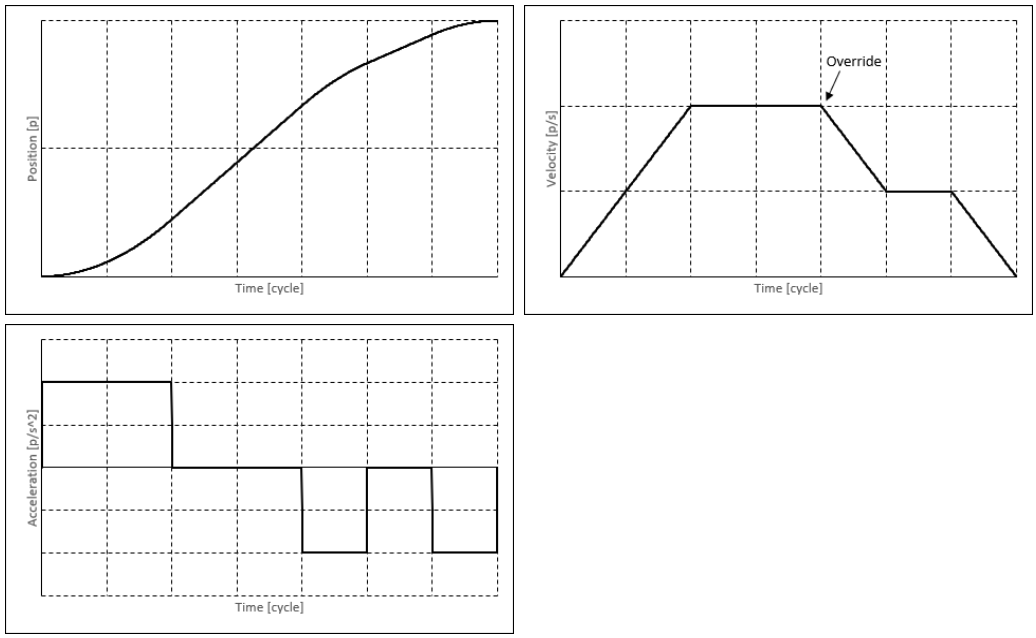


Override From Higher Velocity

오버라이드 시작 시점의 속도가 오버라이드 속도보다 더 크다면, 축은 오버라이드 명령 속도로 감속하게 됩니다. 이 경우 아래 그래프의 위치, 속도, 가감속을 보여줍니다.

오버라이드 모션 명령의 위치, 속도, 가속도

Higher Velocity

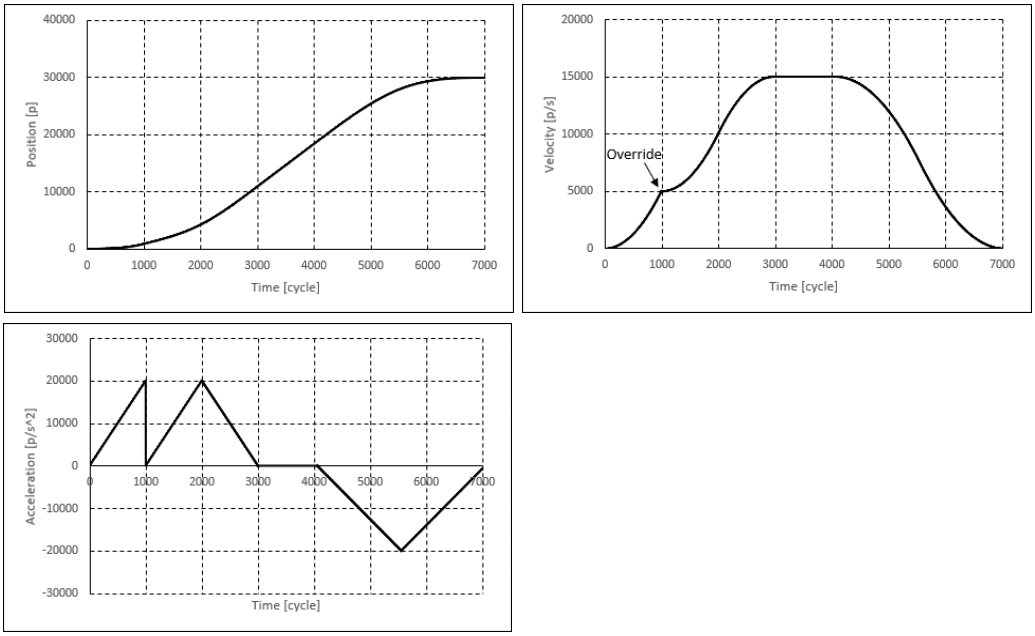


1.6.1.1.4. 오버라이드 가속도

만약, 축의 가감속하는 동안 오버라이드 기능이 수행이 되면, 가속도는 갑자기 0 으로 설정이 되어집니다. 아래의 그래프는 가속구간에 오버라이드가 된 경우의 위치, 속도, 가속도를 보여주고 있습니다. 프로파일 타입은 SCurve 입니다.

오버라이드 가속구간 모션 명령의 위치, 속도, 가속도

Override acceleration

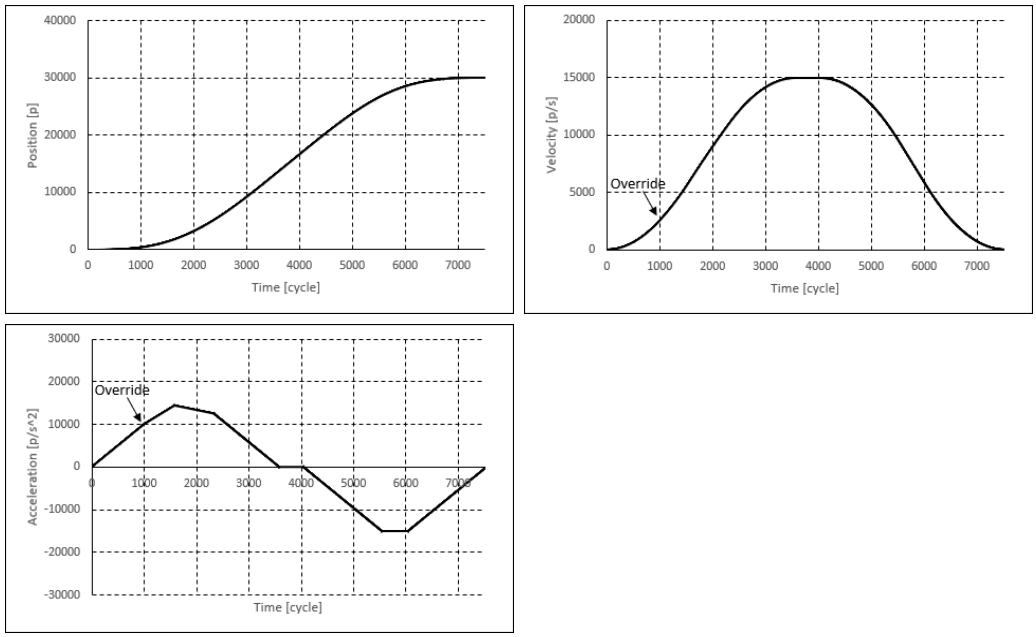


TrapezoidalMAT 프로파일 가속도

TrapezoidalMAT 프로파일은 오버라이드 수행 시 가속도 유지를 위해서 사용됩니다. 다음은 TrapezoidalMAT 프로파일 수행 시 나타나는 위치, 속도, 가속도의 그래프 표를 보여줍니다. moving average time 은 1 로 설정한 경우입니다.

TrapezoidalMAT 프로파일 수행 시 나타나는 위치, 속도, 가속도

Trapezoidal MAT



1.6.1.1.5. 오버라이드 초과이동 방지

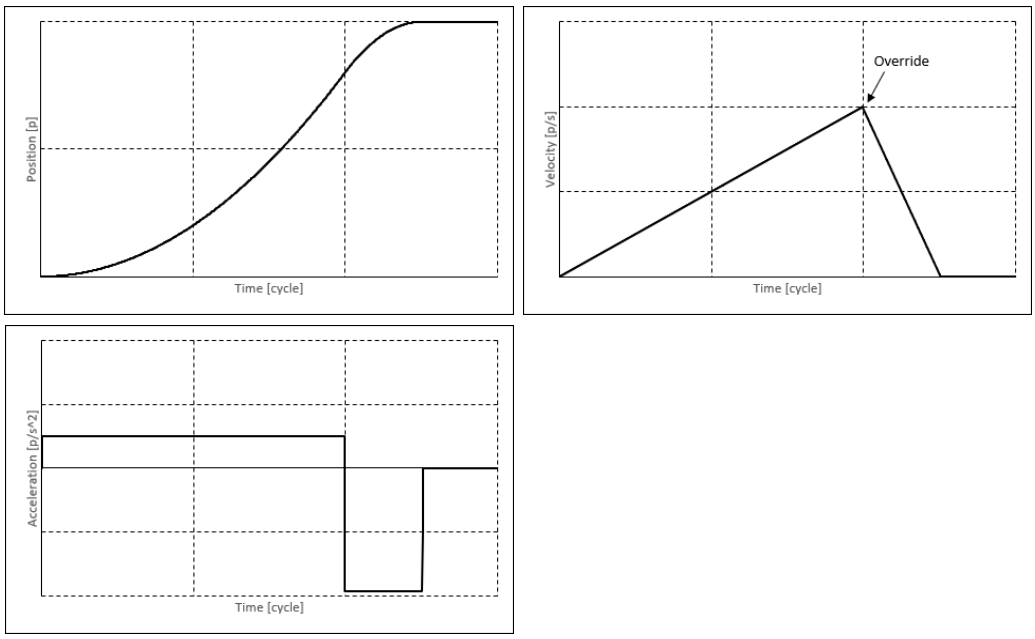
오버라이드 명령을 수행할 때, 축의 속도가 너무 높아서 즉각적은 감속도를 한다 해도 멈출 수 없는 경우가 발생합니다. 이러한 경우를 방지하기 위해 Overtravel 파라미터를 설정하여 프로파일의 모양을 결정할 수 있습니다.

ChangeDeceleration Type

Prohibit Overtravel 파라미터를 ChnageDeceleration 으로 설정한다면, 프로파일의 감속도를 증가시키고 목표위치에서 멈추기 위해 즉각 감속을 시킵니다. 이것은 이 파라미터의 초기값입니다.

OverTravel Parameter 를 ChangeDeceleration 으로 설정 시 위치, 속도, 가감속

Change Deceleration

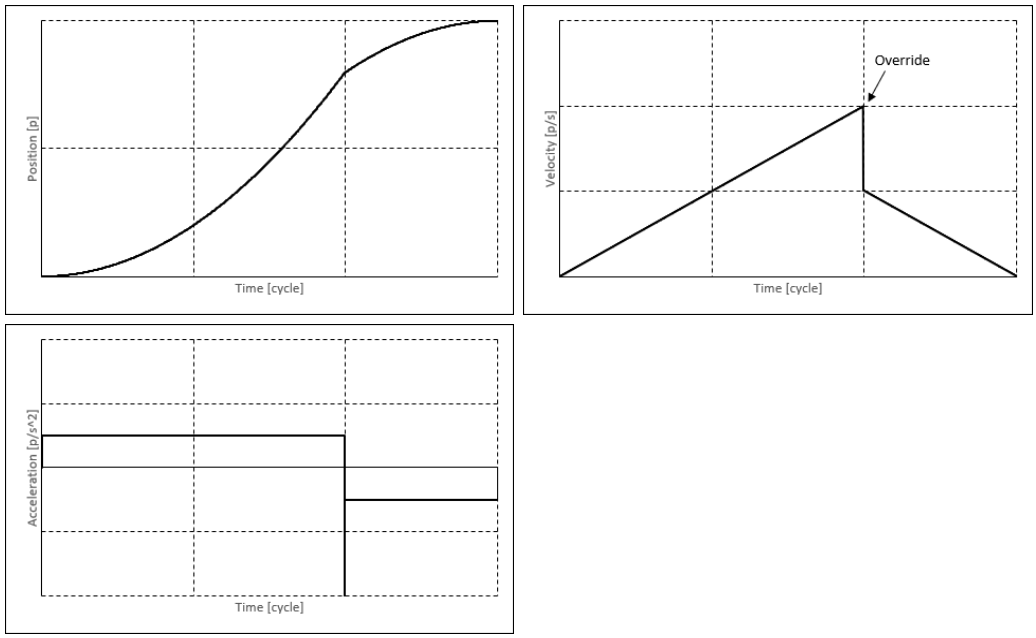


ChangeInitialVelocity Type

만약, Overtravel Parameter 가 ChangeInitialVelocity 로 설정이 되어 있다면, 축이 현재 속도를 바로 감속함으로써 축이 목표위치에서 정지하게 됩니다.

OverTravel Parameter 를 ChangeInitialVelocity 으로 설정 시 위치, 속도, 가감속

Change Initial Velocity

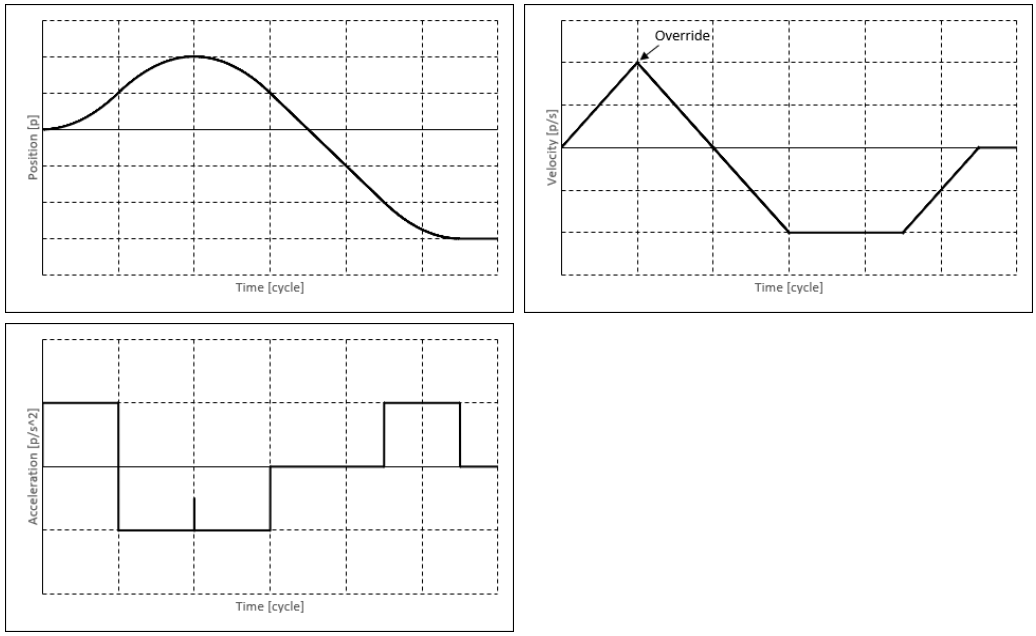


Disabled Type

만약, Prohibit Overtravel 파라미터가 Disabled 로 설정되어 있다면, 축은 즉시 감속을 한다. 그러나 목표 위치에서는 정지하지 않을 수 있습니다. 이 축이 정지한 후, 축은 다시 목표 위치로 오기 위해서는 역방향으로 돌아오게 됩니다.

OverTravel Parameter 를 Disable(비활성)으로 설정 시 위치, 속도, 가감속

Disabled Type

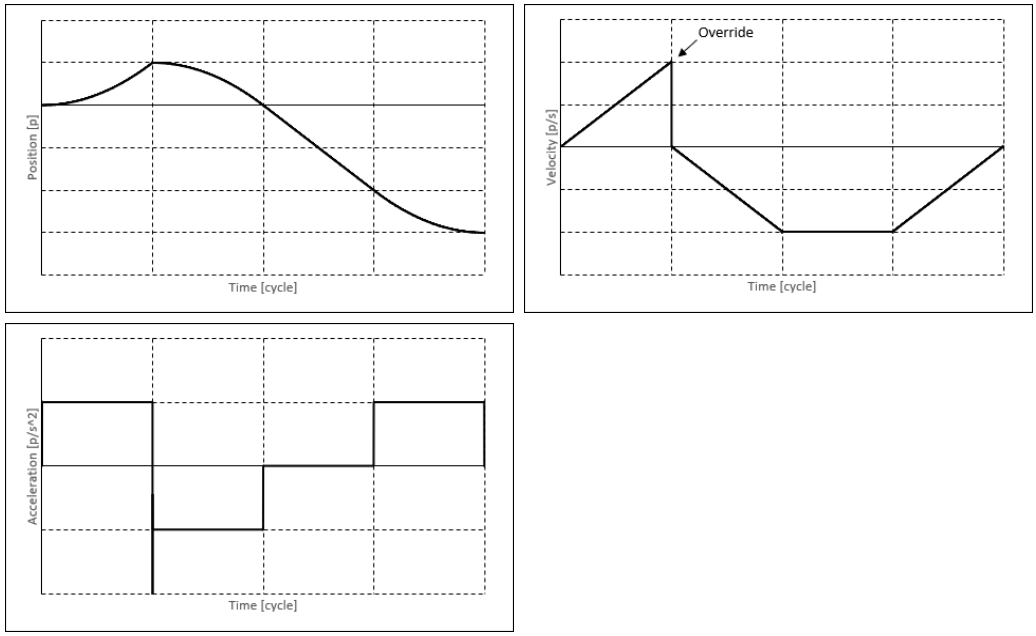


Override in Opposite Direction

오버라이드 목표 위치가 축이 이동하는 방향과 역방향에 존재한다면, 속도는 초과이동을 하지 않기 위해서 0 으로 떨어뜨립니다. (Prohibit overtravel 파라미터가 Disable 로 설정되지 않을 경우) 아래의 그래프는 위치, 속도, 가감속을 보여줍니다. (OverTravel Parameter 가 ChangeDeceleration, ChangeInitialVelocity 설정, 오버라이드 목표위치는 반대방향으로 설정된 상태)

Prohibit overtravel 파라미터가 Disable 로 설정되지 않을 경우 위치, 속도, 가감속

Opposite Direction

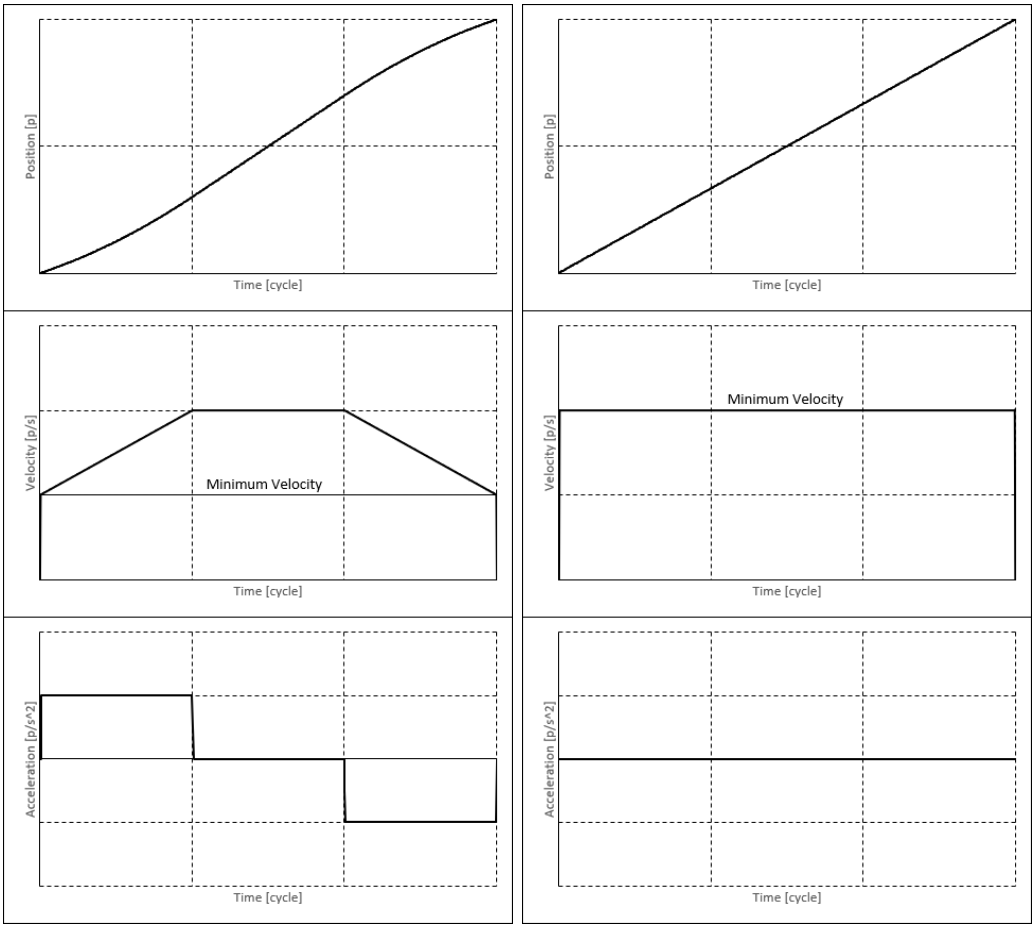


1.6.1.1.6. 잔여 펄스 속도 조정

프로파일을 계산할 때 프로파일 모양과 맞이 않는 여분의 펄스들이 존재합니다. 이러한 여분의 펄스들은 특정 수치의 속도로 감속함에 따라 재 분산을 할 수 있습니다. 이러한 속도 감소는 0 이거나 매우 작습니다. 그러나, 이런 부분은 속도 정확도에 민감한 Application 에 부정적인 영향을 주기에는 충분합니다. 이 경우에, Jerk Ratio Fixed Velocity T, Jerk Ratio Fixed Velocity S, Jerk-Limited Fixed Velocity T, Jerk-Limit Fixed Velocity S 프로파일 타입이 사용돼야만 합니다. 이러한 프로파일 타입은 특정 수치로부터 속도를 감소하지 않고 가감속 구간으로 잔여 펄스의 재배치하게 됩니다.

1.6.1.1.7. 최소 속도

만약, Enable Global Min Velocity 파라미터가 True 로 설정되어 있다면, Global Min Velocity 파라미터 모션프로파일 안에 최소 속도로 명령할 수 있습니다. 만약 시작 속도 및 End 속도를 최소 속도보다 작다면, 그것은 적어도 최소 속도로 설정되게 됩니다. 아래 그래프는 최소속도가 프로파일 속도보다 작게 설정되어 있을 때 위치, 속도, 가속도 값을 보여줍니다.

	최소 속도가 프로파일 속도보다 작은 경우	*최소 속도가 프로파일 속도 보다 큰 경우*
Min Velocity		

1.6.1.2. 모션 프로파일 상세

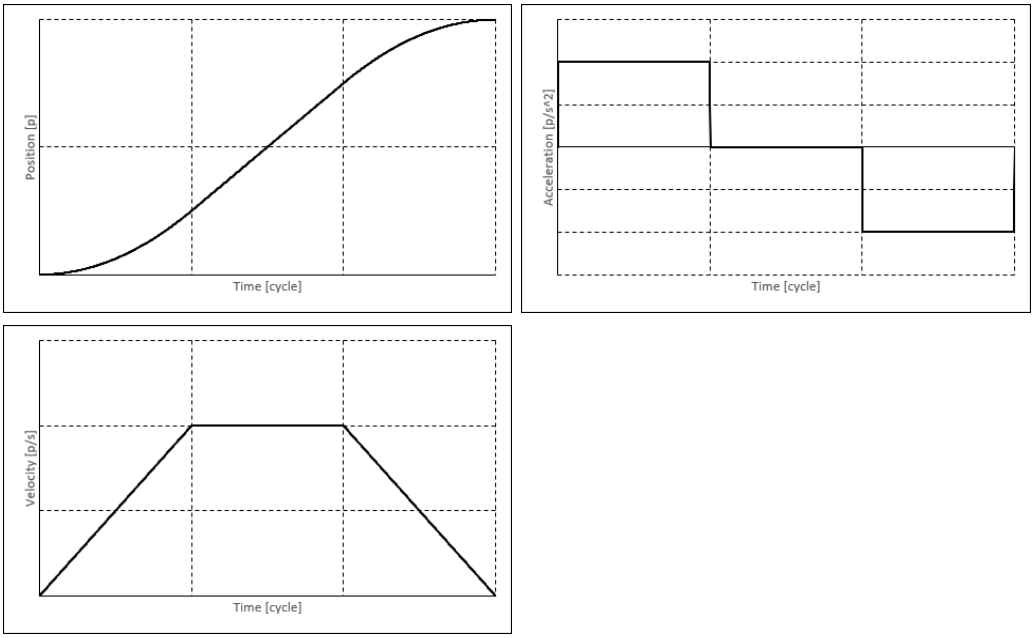
Trapezoidal

Trapezoidal 프로파일은 가장 기본적은 프로파일입니다. 이 프로파일 타입은 정격속도까지 도달하기까지 일정한 가속도를 사용합니다. 프로파일 끝부분에서는 목표위치에서 정지하기 위해서 일정한 감속도를 적용합니다. 이 프로파일은 아래의 파라미터를 사용합니다.

Velocity	Peak velocity
Acc	Peak / average acceleration
Dec	Peak / average deceleration
Starting Velocity	Initial velocity
End Velocity	Final velocity

trapezoidal 프로파일의 위치 가속도

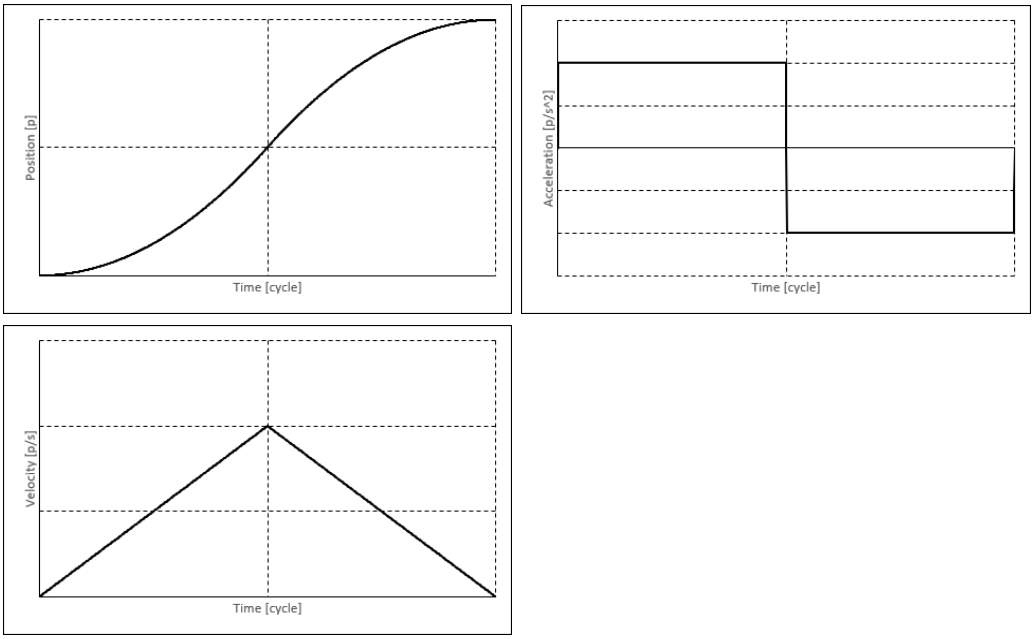
Trapezoidal



만약 목표위치가 설정한 속도 프로파일보다 너무 가까우면 trapezoidal 속도 프로파일은 쐐기모양을 만듭니다. 다음 그래프는 쐐기모양의 위치, 속도, 가감속도를 보여줍니다.

쐐기모양의 위치, 속도, 가감속도

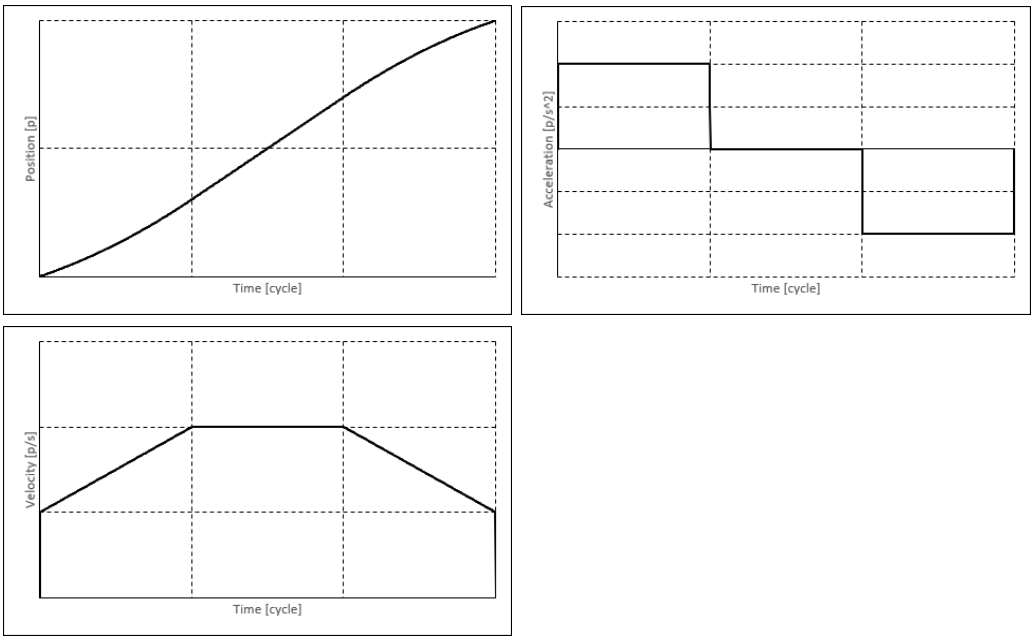
Trapezoidal



만약 초기 및 마지막속도가 설정되어 있다면, 초기 및 마지막에 축의 프로파일 속도까지 도달할 때까지 일정 속도, 가감속도로 시작 마무리할 것입니다. 초기속도, 마지막속도가 프로파일 속도 보다 더 크다면, 프로파일 속도가 맞추어 지게 됩니다. 아래 그래프는 속도, 가감속도 초기, 마지막속도가 지정된 경우의 trapezoidal 프로파일을 보여줍니다.

속도, 가감속도 초기, 마지막속도가 지정된 경우의 trapezoidal 프로파일

Trapezoidal



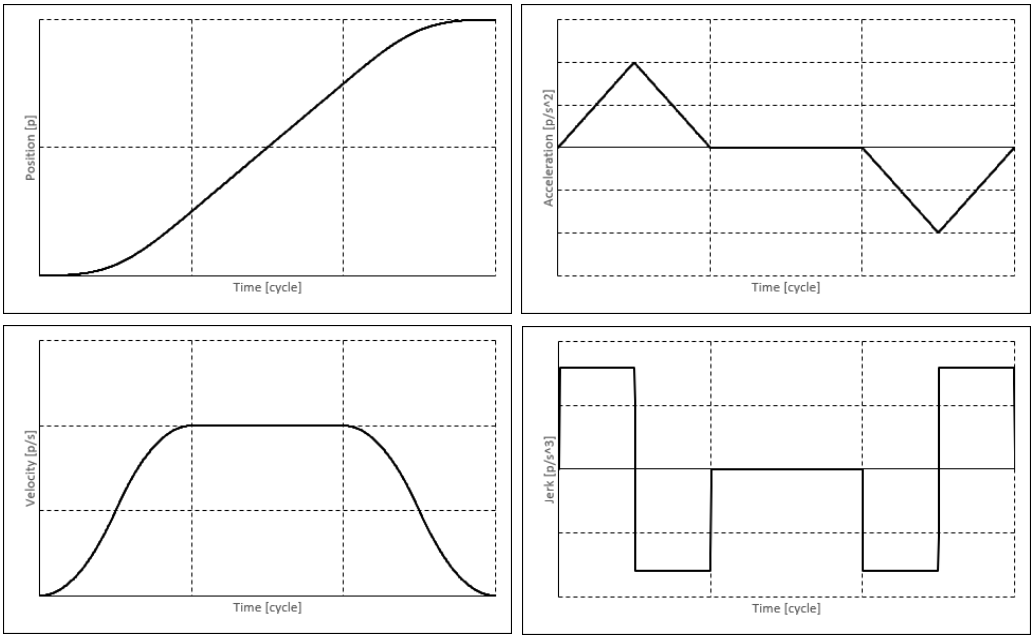
S-Curve

S-Curve 프로파일은 프로파일의 가속도 부분은 두 부분으로 나누어집니다. 첫번째 구간에서는 가속도가 일정 비율로 증가합니다. 그리고 나머지 구간에서는 가속도가 일정 비율로 감소합니다. 이것은 부드럽게 S 모양의 가속도 곡선을 형성하며 감속도도 이와 비슷합니다. Trapezoidal 프로파일과 비교해서 만약, 가속도 시간이 서로 같다면, 최대 가속 수치는 2 배의 수치가 됩니다. S-Curve 는 다음 프로파일 파라미터를 사용하며 아래 그래프는 S-커브의 위치, 속도, 가속도 저크를 나타냅니다.

Velocity	Peak velocity
Acc	Average acceleration
Dec	Average deceleration
Starting Velocity	Initial velocity
End Velocity	Final velocity

S-커브의 위치 속도, 가속도 저크

S-Curve



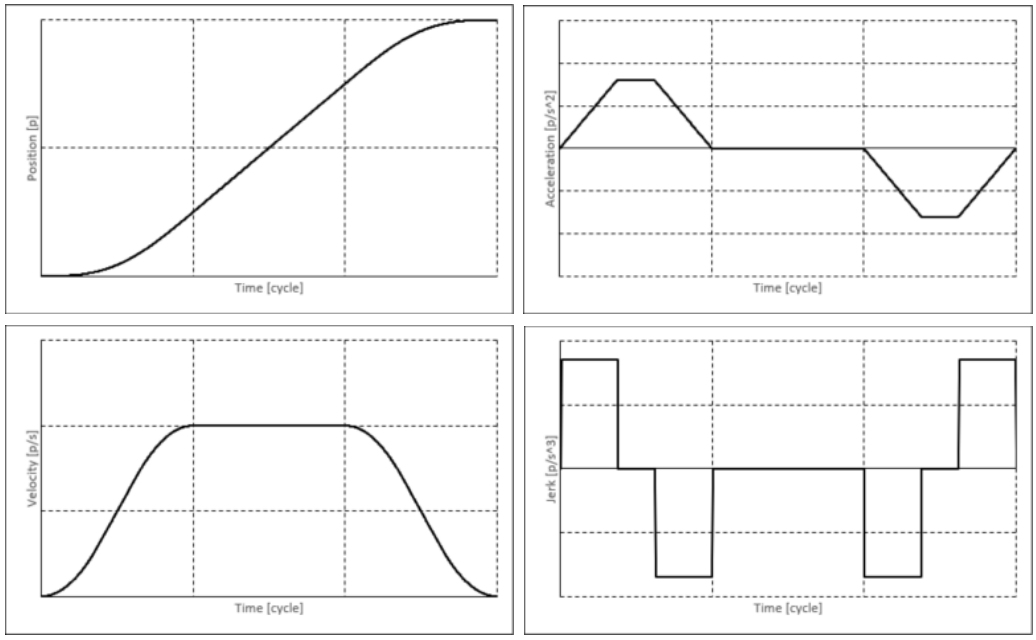
Jerk Ratio

Jerk Ratio 프로파일은 Trapezoidal 과 S-Curve 사이의 특성을 보여줍니다. 가속 저크 비율은 가속도 곡선의 모양을 결정합니다. 가속도가 변화되어지는 시간의 비율을 결정합니다. (저크 값이 0 이 아닌 경우), 감속도 저크 비율은 감속도 곡선 모양을 결정합니다. 가감속도 저크 비율은 0과 1 사이의 수치 값입니다. 만약, 가감속도 저크 비율이 0이면, Trapezoidal 프로파일을 됩니다. 만약 가속도 저크 비율이 1이면, 프로파일은 S-Curve 가 됩니다. 이 프로파일은 아래의 파라미터 값을 따르게 됩니다.

Velocity	Peak velocity
Acc	Average acceleration
Dec	Average deceleration
Jerk Acc Ratio	Acceleration jerk ratio
Jerk Dec Ratio	Deceleration jerk ratio
Starting Velocity	Initial velocity
End Velocity	Final velocity

저크 비율이 0.75 설정되었을 때 위치, 가감속

Jerk Ratio



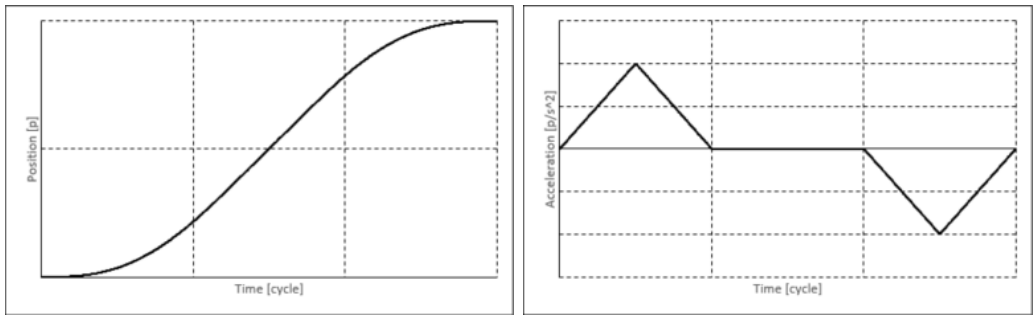
Parabolic

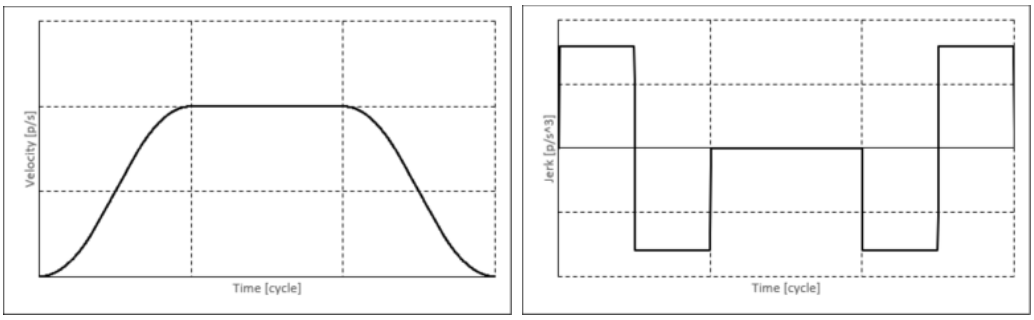
Parabolic 프로파일은, 축의 가감속도 값은 가감속 저크 값이 선형적으로 가감속에 적용되며 아래의 파라미터를 사용합니다.

Velocity	Peak velocity
Acc	Average acceleration
Dec	Average deceleration
Starting Velocity	Initial velocity
End Velocity	Final velocity

Parabolic 프로파일 위치, 속도, 가속도, 저크

Parabolic





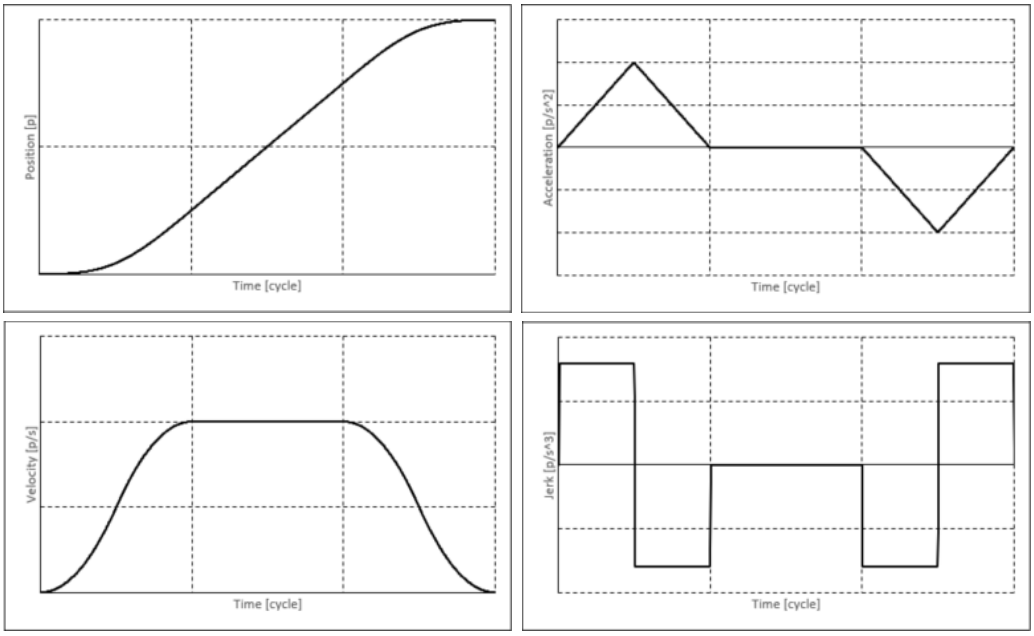
Sine

Sine 프로파일은, 축의 가감속도가 sin 곡선을 형성합니다. 이 프로파일은 아래의 파라미터 설정을 사용합니다.

Velocity	Peak velocity
Acc	Peak acceleration
Dec	Peak deceleration
Starting Velocity	Initial velocity
End Velocity	Final velocity

Sine 프로파일의 위치, 속도, 가감속, 저크

Sine



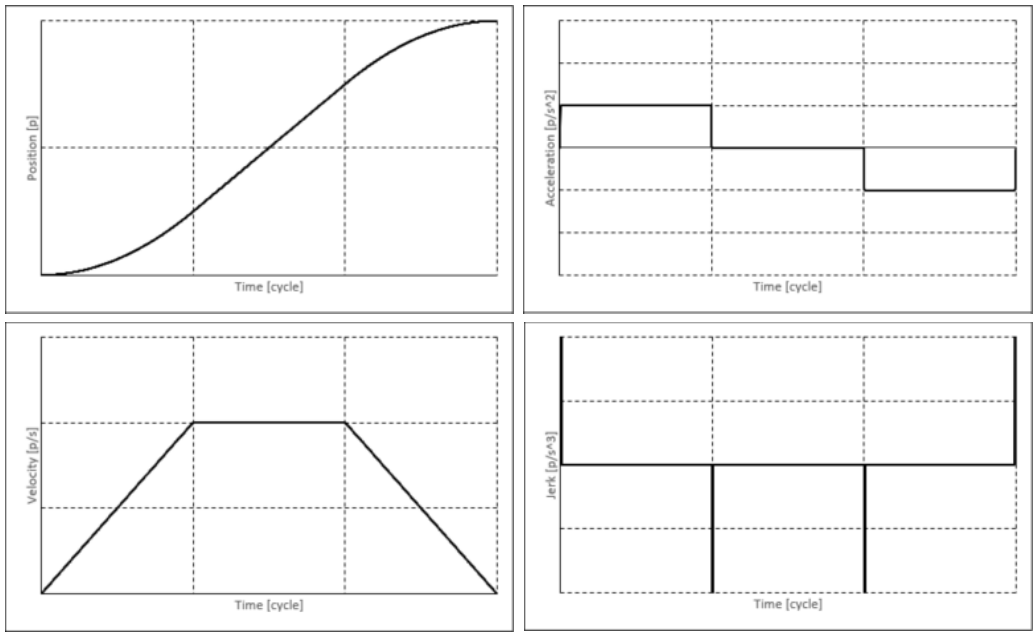
Advanced-S

Advanced-S 프로파일에서 가속 저크와 감속 저크는 선형적으로 변경됩니다. 가속 및 가속 저크는 프로파일 시작 부분에서 모두 0 이며 프로파일 전체에서 연속적입니다. 또한 가속 저크 비율(및 감속 저크 비율)을 지정하여 가속이 변경되는 시간(저크는 0 이 아님)과 총 가속 시간의 비율을 결정할 수 있습니다. 가속 저크 비와 감속 저크 비율은 0 과 1 사이의 값이어야 합니다. 저크 비율이 높을수록 피크 가속도가 높아지고 낮은 저크 피크 값이 됩니다. 이 프로파일은 아래의 파라미터를 필요합니다.

Velocity	Peak velocity
Acc	Average acceleration
Dec	Average deceleration
Jerk Acc Ratio	Acceleration jerk ratio
Jerk Dec Ratio	Deceleration jerk ratio
Starting Velocity	Initial velocity
End Velocity	Final velocity

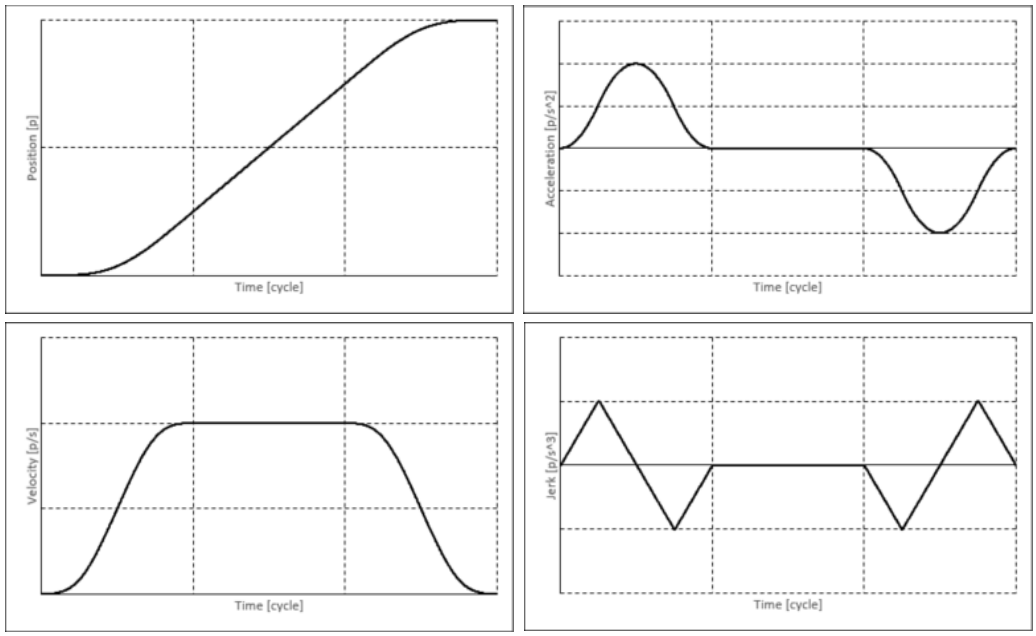
저크 비율이 0 인 경우 프로파일의 위치, 속도 가감속도 저크(Trapezoidal 프로파일 같음)

Advanced-S



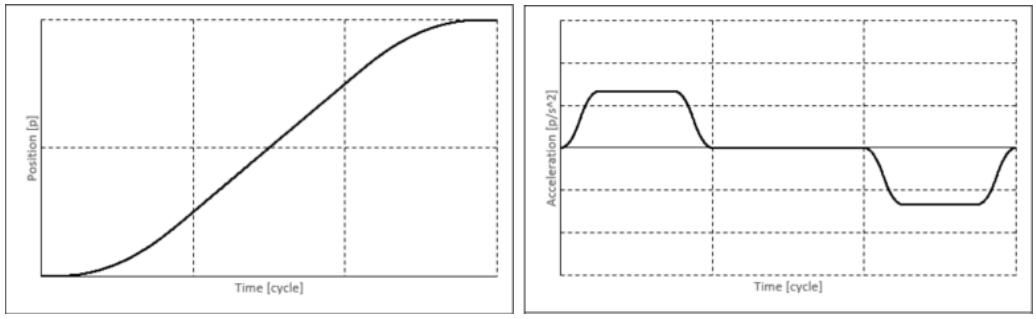
저크 비율이 1 인 경우 프로파일의 위치, 속도 가감속도 저크

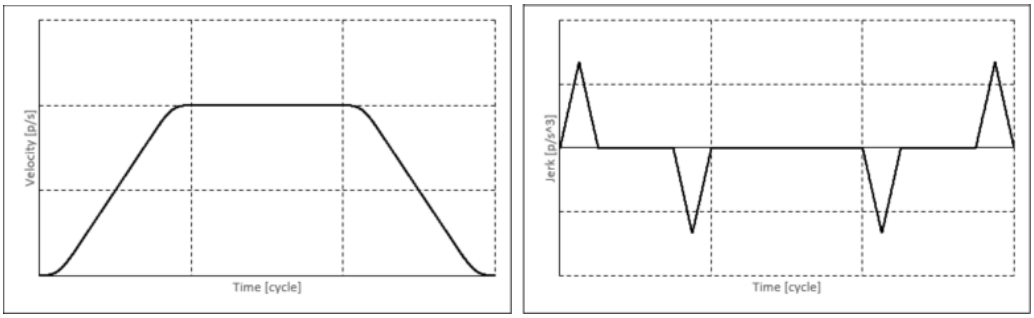
Advanced-S



저크 비율이 0.5 인 경우 프로파일의 위치, 속도 가감속도 저크

Advanced-S





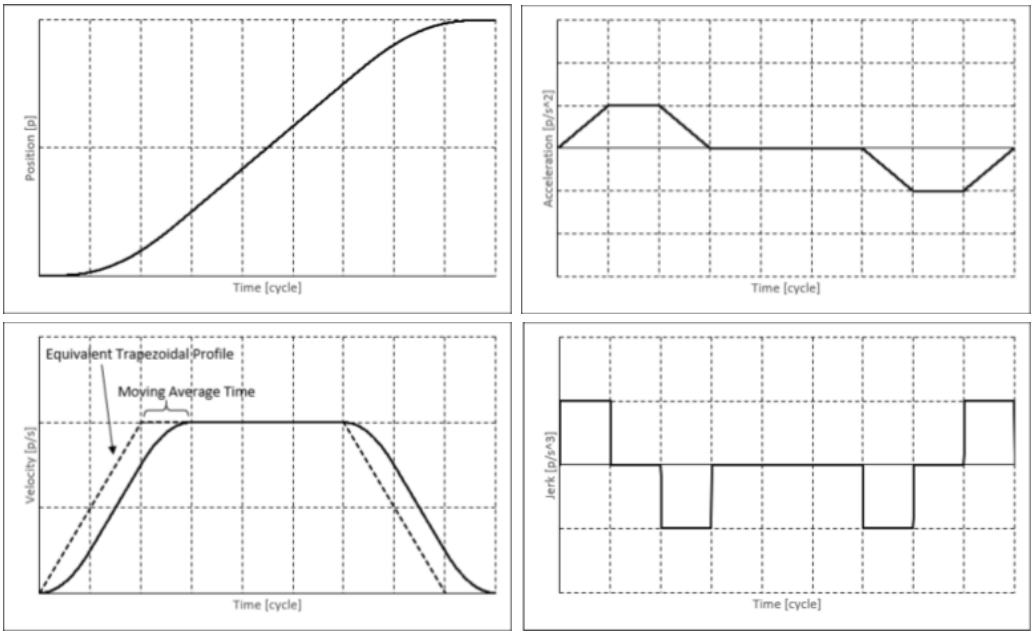
Trapezoidal Moving Average Time

Trapezoidal Moving Average Time 프로파일은 Trapezoidal 프로파일과 같습니다. 그러나 갑작스러운 가속도의 변화를 방지하기 위해 이동 평균 필터의 평균시간을 적용하였습니다. 이동 평균 필터의 평균 시간은 이동 평균 시간 파라미터에 의해 결정됩니다. 이 수치는 여러 사이클 타임에서 가장 가깝게 수치로 결정됩니다. Moving Average Time 이 0 이다면 이 프로파일은 Trapezoidal 프로파일이 됩니다. 프로파일과 비교하여, 프로파일 이동 평균 시간은 정해진 수치로 감속합니다. 만약 프로파일 높은 초기 및 마지막 속도에 의해 만들어 줄 수 없다면, Moving Average time 은 특정 수치로 감속하게 됩니다. 이 프로파일은 아래의 파라미터를 이용합니다.

Velocity	Peak velocity
Acc	Peak acceleration
Dec	Peak deceleration
Starting Velocity	Initial velocity
End Velocity	Final velocity
Moving Average Time Milliseconds	Moving average time in milliseconds

Trapezoidal moving average 프로파일의 위치, 속도 가감속도 저크

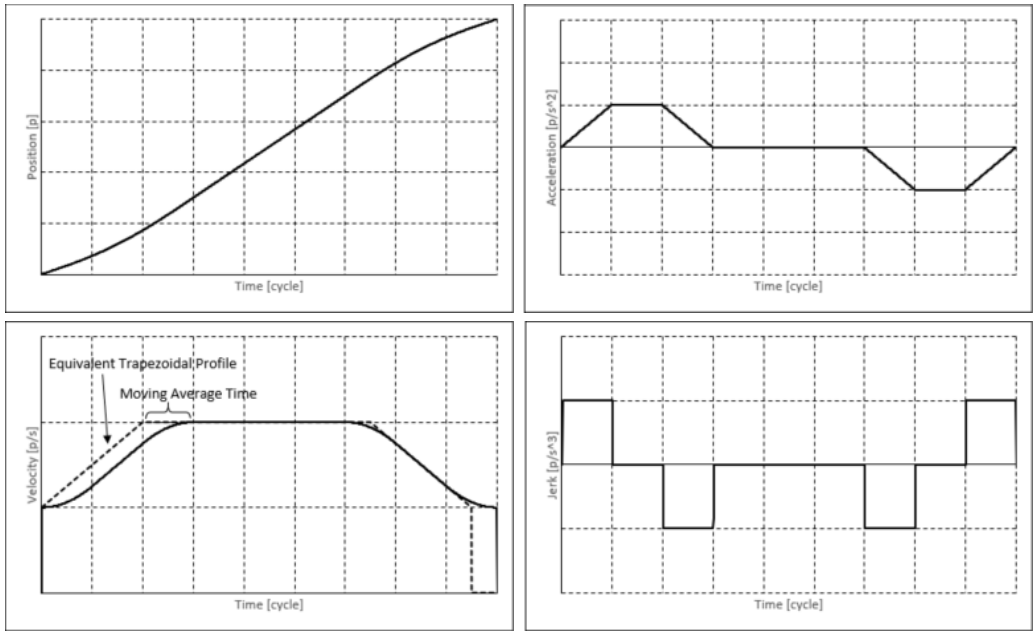
Trapezoidal Moving Average Time



다른 프로파일과 비슷하게 Trapezoidal moving average time 프로파일은 초기 및 마지막 속도의 설정 및 오버라이드 기능을 지원합니다. 초기 마지막 속도가 0 이 아닌 경우, 프로파일은 같은 가감속도 설정으로 이동경로 정확한 초기 마지막 속도를 유지하도록 조정합니다.

초기 마지막 속도가 0 이 아닌 경우 Trapezoidal moving average 프로파일의 위치, 속도 가감속도 저크

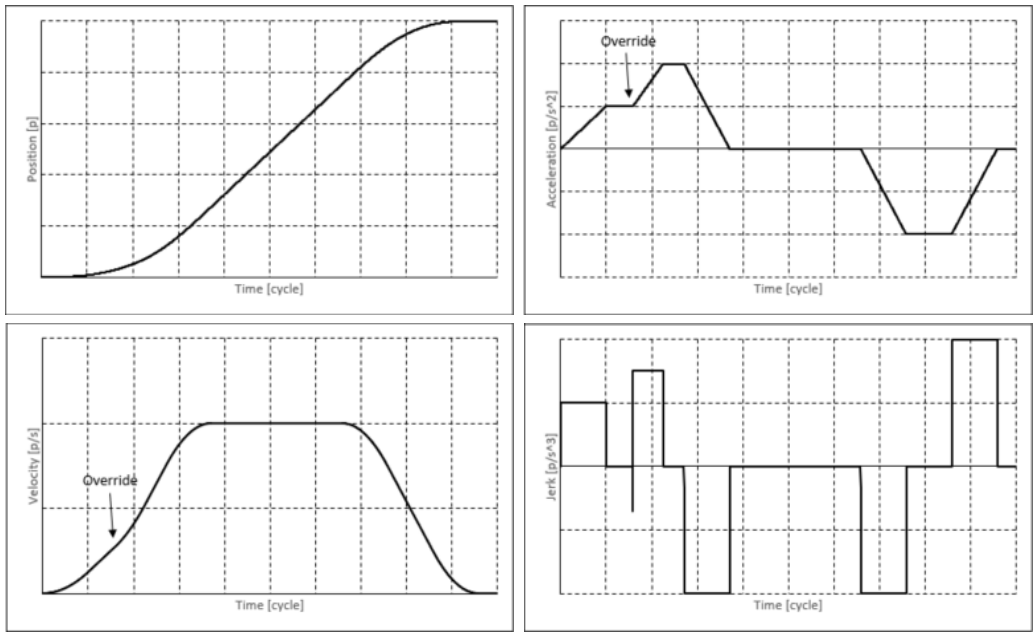
Trapezoidal Moving Average Time



Trapezoidal moving average time (사다리꼴 이동 평균 시간) 프로파일은 오버라이드 시작 시 가속을 0 으로 설정하는 대신 오버라이드 중에 가속을 보존한다는 점에서 다른 WMX3 프로필 유형과 다릅니다. 이를 통해 축이 가속 또는 감속하는 동안 오버라이드가 실행될 때 기계의 물리적 부담을 줄일 수 있습니다.

축이 가속하는 동안에 Trapezoidal moving average time 프로파일의 위치, 속도, 가속도

Trapezoidal Moving Average Time



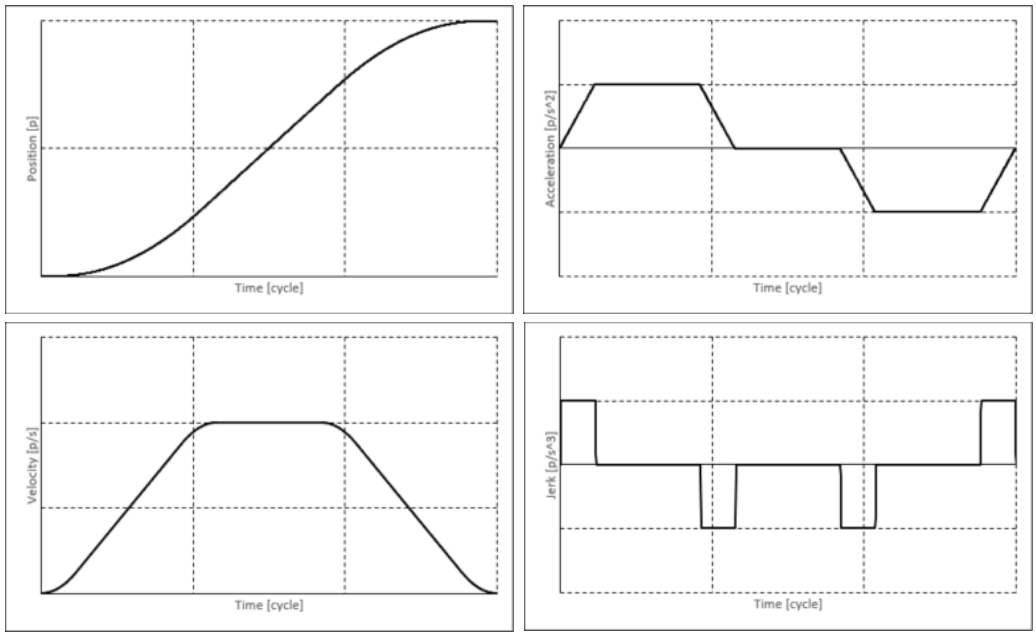
Jerk-Limited

Jerk-Limited 프로파일은 가감속도 및 가감속 저크의 값을 제한할 수 있는 가장 빠른 프로파일입니다. 일반적으로 Jerk-Limited 프로파일은 Trapezoidal 과 S-Curve 프로파일 사이의 특징을 보입니다. 이 프로파일은 아래의 파라미터를 사용합니다.

Velocity	Peak velocity
Acc	Peak acceleration
Dec	Peak deceleration
Jerk Acc Ratio	Peak acceleration jerk
Jerk Dec Ratio	Peak deceleration jerk
Starting Velocity	Initial velocity
End Velocity	Final velocity

Jerk-limited 프로파일의 위치, 속도, 가속도, 제한 저크

Jerk-Limited



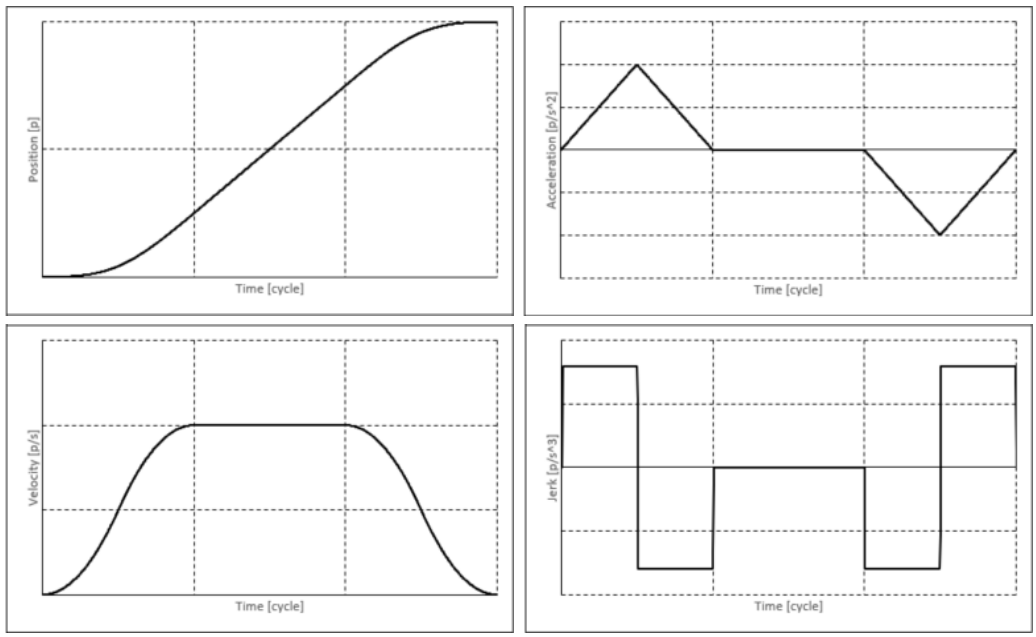
Jerk-Limited S-Curve

Jerk-Limited S-Curve 프로파일은, 가속 구간을 두 부분으로 나누어집니다. 첫 구간에서는 가속도 값이 증가하고 나머지 구간에서는 가속도 값이 일정 비율로 감속합니다. (이 때 가속도, 가속도 저크 제한을 수행되지 않습니다.) 감속도 곡선은 일정한 감속도, 감속 저크 제한 값에 의해 유사하게 나타납니다. Jerk-Limited S-Curve 프로파일과 S-Curve 프로파일 사이의 차이점은 가감속 저크로 설정이 되어있고 가감속은 평균값 대신 제한 값이 적용이 된다는 것입니다. 이 프로파일은 아래의 파라미터를 이용합니다.

Velocity	Peak velocity
Acc	Peak acceleration
Dec	Peak deceleration
Jerk Acc	Peak acceleration jerk
Jerk Dec	Peak deceleration jerk
Starting Velocity	Initial velocity
End Velocity	Final velocity

Jerk-limited S-Curve 프로파일의 위치, 속도, 가속도, 저크

Jerk-Limited S-Curve



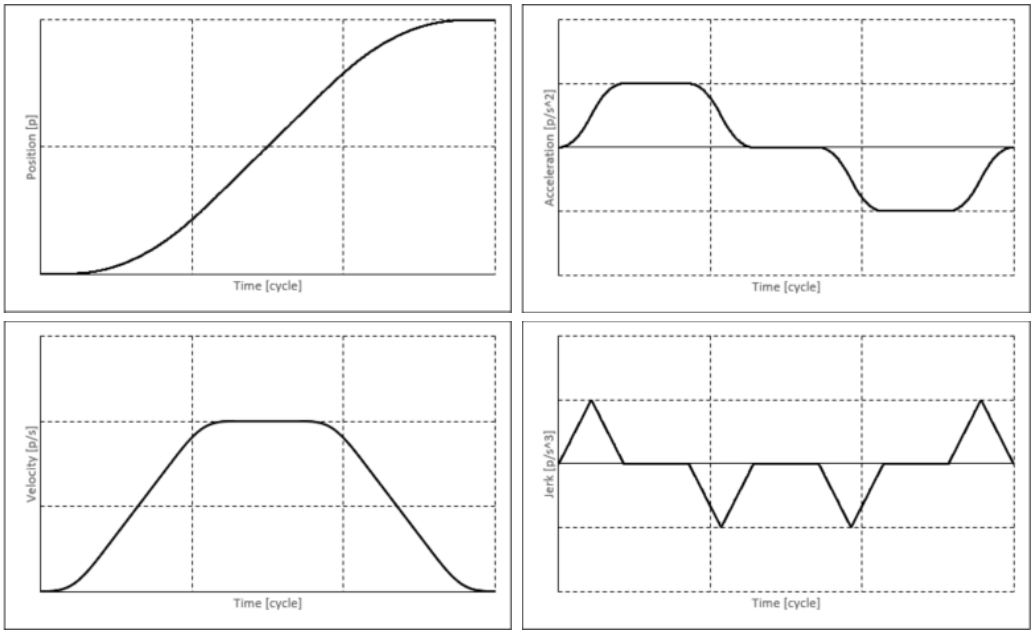
Jerk-Limited Advanced-S

Jerk-Limited Advanced-S 프로파일은 가감속 저크값이 선형적으로 변화합니다. 가속도, 가속 저크값은 시작 지점이 둘 다 0 이면 프로파일 구간동안 연속적인 모양을 만듭니다. 더불어 가속도 및 가속 저크값은 감속 동안에 감속 및 감속 저크값의 제한을 받게 됩니다. 이 프로파일은 아래의 파라미터를 이용합니다.

Velocity	Peak velocity
Acc	Peak acceleration
Dec	Peak deceleration
Jerk Acc	Peak acceleration jerk
Jerk Dec	Peak deceleration jerk
Starting Velocity	Initial velocity
End Velocity	Final velocity

Jerk-limited Advanced S-Curve 프로파일의 위치, 속도, 가속도, 저크

Jerk-Limited Advanced-S



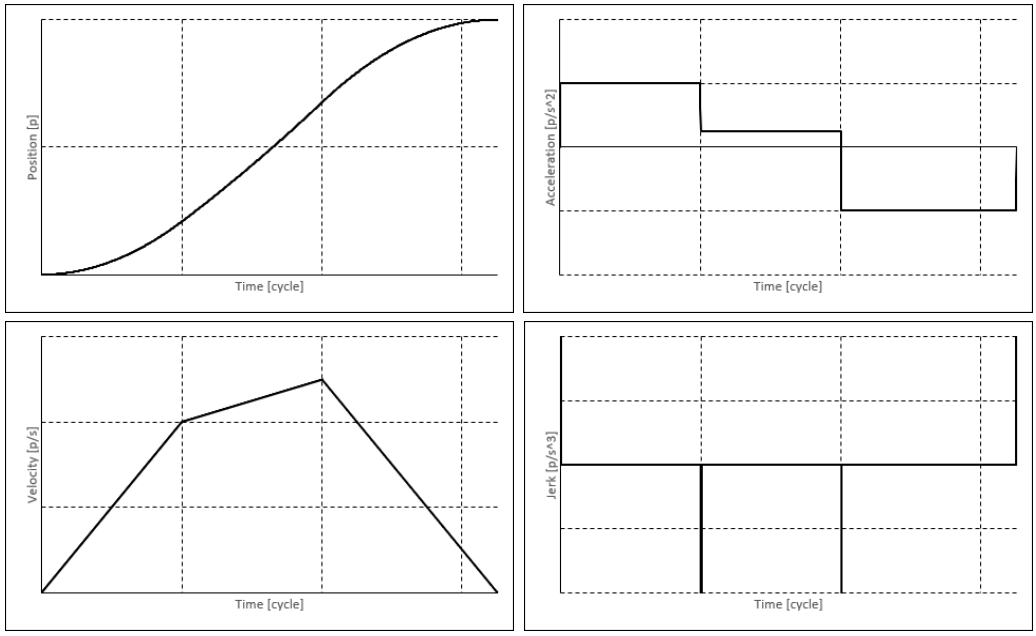
Two Velocity Trapezoidal

Two-Velocity Trapezoidal 프로파일은 설정 속도에 도달할 때까지 일정한 가속도로 가속을 하게 된다. 설정 속도에 도달한 후, 두번째 설정속도까지 일정하게 다시 가속을 하게 됩니다. 두번째 설정속도에 도달한 후, 축은 목표위치에서 멈추기 위해서 일정 비율로 감속하게 되어집니다. 중간 가속도 값은 첫 속도에서 두번째 속도로 자동 계산된 값으로 가속하게 됩니다. 중간 가속도 값은 프로파일 가속도 값을 초과할 수 없습니다. 만약 목표 위치가 너무 짧아서 두번째 속도로 설정된 가감속도를 이용하여 도달하지 못한다면 축은 두번째 속도에 도달하기 전에 감속을 시작합니다. 첫 속도 값은 두번째 속도 값보다 작아야 합니다. 만약 첫 속도 값이 두번째 속도보다 크다면, 첫 속도는 두번째 값으로 설정을 하게 됩니다. 두번째 속도 프로파일은 조그 및 속도 명령어와 호환이 됩니다.

Velocity	First target velocity
Acc	Peak / average acceleration
Dec	Peak / average deceleration
Jerk Acc	Peak acceleration jerk
Jerk Dec	Peak deceleration jerk
Starting Velocity	Initial velocity
End Velocity	Final velocity
Second Velocity	Second target velocity

Two-Velocity Trapezoidal 프로파일의 위치, 속도, 가속도, 저크

Two-Velocity Trapezoidal



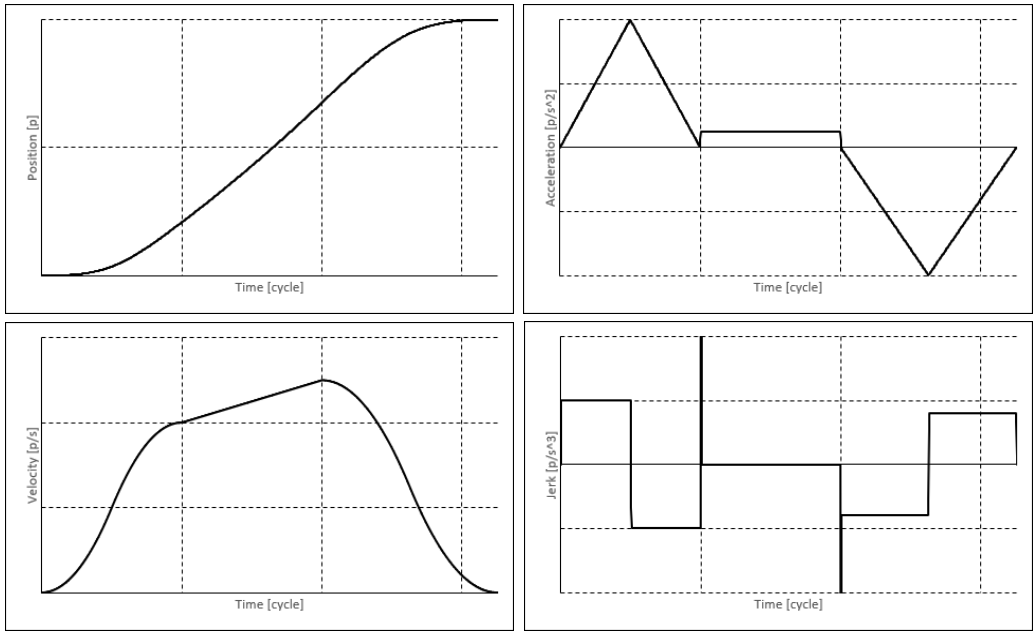
Two Velocity S-Curve

Two-Velocity S-Curve 프로파일, 프로파일 가속 곡선은 두 부분으로 나누어집니다. 첫 구간은 가속구간은 일정 비율로 증가를 합니다. 두번째 구간에서는 가속 구간은 일정 비율로 감소합니다. 이것은 S 모양의 가속 곡선을 형성합니다. 축이 설정 속도에 도달한 후에, 축은 두번째 속도에 도달하기까지 중간 가속도로 가속이 되어집니다. 두번째 속도에 도달한 후 축은 S 모양의 감속 곡선을 형성합니다. 가속 구간과 비슷합니다. 중간 가속도의 값은 첫 설정 속도에서 두번째 설정속도로 자동 계산이 되어집니다. 중간 가속도 값은 프로파일 가속도 값을 초과할 수 없습니다. 만약 목표 위치가 너무 짧아서 두번째 속도로 설정된 가감속도를 이용하여 도달하지 못한다면 축은 두번째 속도에 도달하기 전에 감속을 시작합니다. 첫 속도 값은 두번째 속도 값보다 작아야 합니다. 만약 첫 속도 값이 두번째 속도보다 크다면, 첫 속도는 두번째 값으로 설정을 하게 됩니다. 두번째 속도 프로파일은 조그 및 속도 명령어와 호환이 됩니다.

Velocity	First target velocity
Acc	Average acceleration
Dec	Average deceleration
Starting Velocity	Initial velocity
End Velocity	Final velocity
Second Velocity	Second target velocity

Two-Velocity S-curve 프로파일의 위치, 속도, 가속도, 저크

Two-Velocity S-curve



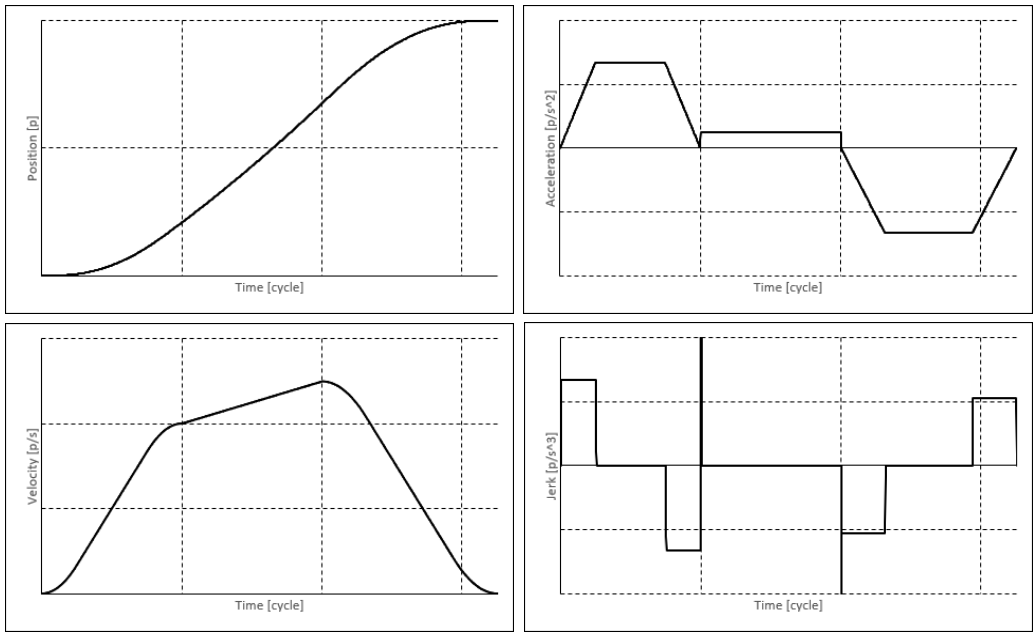
Two Velocity Jerk Ratio

Two-Velocity Jerk Ratio 프로파일은 가속 곡선의 모양은 가속 저크 비율에 의해 결정이 됩니다. 저크 값이 0 이 아닌 경우 가속은 총 가속 구간 시간에 실제 감속 시간 비율에 의해 결정됩니다. 축의 설정속도로 가속한 후에 축은 저크 비율로 감속을 사용하여 감속을 하게 됩니다. 가속 및 감속 저크 비율은 0 과 1 사이에 있어야 합니다. 중간 가속도의 값은 첫 설정 속도에서 두번째 설정속도로 자동 계산이 되어집니다. 중간 가속도 값은 프로파일 가속도 값을 초과할 수 없습니다. 만약 목표 위치가 너무 짧아서 두번째 속도로 설정된 가감속도를 이용하여 도달하지 못한다면 축은 두번째 속도에 도달하기 전에 감속을 시작합니다. 첫 속도 값은 두번째 속도 값보다 작아야 합니다. 만약 첫 속도 값이 두번째 속도보다 크다면, 첫 속도는 두번째 값으로 설정을 하게 됩니다. 두번째 속도 프로파일은 조그 및 속도 명령어와 호환이 됩니다. 이 프로파일은 아래의 파라미터를 이용합니다.

Velocity	First target velocity
Acc	Average acceleration
Dec	Average deceleration
Jerk Acc Ratio	Acceleration jerk ratio
Jerk Dec Ratio	Deceleration jerk ratio
Starting Velocity	Initial velocity
End Velocity	Final velocity
Second Velocity	Second target velocity

Two-Velocity Jerk Ratio 프로파일의 위치, 속도, 가속도, 저크 (저크 비율 0.5)

Two-Velocity Jerk Ratio



Time Acceleration Trapezoidal

Time Acceleration Trapezoidal 프로파일은 가감속 부분을 User Unit per sec^2 대신 시간 단위로 설정한 Trapezoidal 프로파일로 아래의 파라미터를 이용합니다.

Velocity	Peak velocity
Acceleration Time	Acceleration time in milliseconds
Deceleration Time	Deceleration time in milliseconds
Starting Velocity	Initial velocity
End Velocity	Final velocity

Time Acceleration S-Curve

Time Acceleration S-Curve 프로파일은 가감속 부분의 단위가 User Unit per sec^2 대신 시간 단위로 설정한 S-Curve 프로파일입니다. 이 프로파일은 아래의 파라미터를 이용합니다.

Velocity	Peak velocity

Acceleration Time	Acceleration time in milliseconds
Deceleration Time	Deceleration time in milliseconds
Starting Velocity	Initial velocity
End Velocity	Final velocity

Time Acceleration Jerk Ratio

Time Acceleration Jerk Ratio 프로파일은 가감속 부분의 단위가 User Unit per sec^2 대신 시간 단위로 설정된 Jerk Ratio 프로파일입니다.

Velocity	Peak velocity
Acceleration Time	Acceleration time in milliseconds
Deceleration Time	Deceleration time in milliseconds
Jerk Acc Ratio	Acceleration jerk ratio
Jerk Dec Ratio	Deceleration jerk ratio
Starting Velocity	Initial velocity
End Velocity	Final velocity

Time Acceleration Parabolic

Time Acceleration Parabolic 프로파일은 가감속 부분의 단위가 User Unit per sec^2 대신 시간 단위로 설정된 Parabolic 프로파일입니다.

Velocity	Peak velocity
Acceleration Time	Acceleration time in milliseconds
Deceleration Time	Deceleration time in milliseconds
Starting Velocity	Initial velocity
End Velocity	Final velocity

Time Acceleration Sine

Time Acceleration Sine 프로파일은 가감속 부분의 단위가 UserUnit per sec^2 대신 시간 단위로 설정된 Sine 프로파일로 이 프로파일은 아래의 파라미터를 이용합니다.

Velocity	Peak velocity
Acceleration Time	Acceleration time in milliseconds
Deceleration Time	Deceleration time in milliseconds
Starting Velocity	Initial velocity
End Velocity	Final velocity

Time Acceleration Advanced-S

Time Acceleration Advanced-S 프로파일은 가감속 부분의 단위가 UserUnit per sec^2 대신 시간 단위로 설정된 Advanced-S 프로파일입니다. 이 프로파일은 아래의 파라미터를 이용합니다.

Velocity	Peak velocity
Acceleration Time	Acceleration time in milliseconds
Deceleration Time	Deceleration time in milliseconds
Jerk Acc Ratio	Acceleration jerk ratio
Jerk Dec Ratio	Deceleration jerk ratio
Starting Velocity	Initial velocity
End Velocity	Final velocity

1.7. 라이브러리

WMX3 는 C++ (32, 64 비트), .NET 라이브러리 (C#, VB.NET, etc...), C++ Builder (32, 64 비트)와 Python 용 라이브러리를 제공합니다.

	WMX Ver.	C++	.NET	Python	C++ Builder
Library	v3.3 v3.4u4	VC++ 2012, 2013, 2015, 2017 + 2019	.NET 4.0 and higher	3.6	XE7

1.7.1. 라이브러리 개요

WMX3 의 C++, .NET, Python, C++ Builder, RTX 응용프로그램 라이브러리 개요를 설명합니다. 사용하는 라이브러리에 맞추어 아래의 내용을 참고하시기 바랍니다.

1.7.1.1. C++

C++은 WMX3 에서 지원하는 기본 언어입니다. C++ 어플리케이션을 작성한 다음 기본 API 라이브러리 (.lib)를 링크 하여 라이브러리 함수를 호출하여 WMX3 엔진에 명령을 보낼 수 있습니다.

	C++ Library
라이브러리 파일	✓ C++ 라이브러리는 설치 디렉토리의 Lib 폴더에 설치됩니다. ✓ C++ 헤더는 설치 디렉토리의 Include 폴더에 설치됩니다. ✓ C++ 라이브러리 및 헤더 파일 목록은 1.7.2. 라이브러리 목록 페이지를 참조하세요.
x86 라이브러리	WMX3 64-bit 설치에는 x64 와 x86 모두 C++ 라이브러리를 포함하고 있습니다. 추가 정보는 1.7.3.3. WMX3 x64 → x86 라이브러리 페이지를 참조하십시오.
사용 방법	✓ C++ 라이브러리 사용 방법은 ------- 프로젝트 만들기 페이지를 참조하세요. ✓ C++ 라이브러리를 사용하는 샘플 프로젝트 목록은 ------ 참조하세요.
통합 개발 환경	Microsoft Visual Studio 2012, 2013, 2015, 2017, 2019
*Note	✓ 개발 환경 지원사항은 WMX3 버전에 따라 상이 할 수 있습니다. 자세한 사항은 1.7 라이브러리 항목을 참고하시기 바랍니다. ✓ Microsoft Visual Studio 2015 이상을 사용하는 경우 1.7.2 라이브러리 주의 사항 항목 내용의 Visual Studio 2015 이상 사용자를 알림 페이지에서 설명한대로 특정 Microsoft 라이브러리가 프로젝트와 연결되어 있어야 합니다.

1.7.1.2. .NET

C++ 라이브러리의 거의 모든 함수는 .NET 라이브러리에서도 사용할 수 있지만 일부 특수 기능은 사용할 수 없습니다. C#, VB.NET, C++, CL 같은 언어를 사용할 수 있으며 .NET 라이브러리의 인터페이스는 가능한 C++ 라이브러리와 유사하게 만들어집니다. 예를 들어 모든 기능 이름과 매개 변수 이름은 동일하게 유지됩니다. 따라서 .NET 개발자는 C++용 API 를 참조하시기 바랍니다.

	.NET Library
라이브러리 파일	✓ NET 라이브러리는 설치 디렉토리의 Lib 폴더에 설치됩니다. ✓ .NET 라이브러리 파일 목록은 1.7.2. 라이브러리 목록 페이지를 참조하세요.
사용 방법	✓ .NET 라이브러리 사용 방법은 ------- 프로젝트 만들기 페이지를 참조하세요. ✓ C# 라이브러리를 사용하는 샘플 프로젝트 목록은 ------ 참조하세요.
통합 개발 환경	Microsoft Visual Studio 2012, 2013, 2015, 2017, 2019 :: LabVIEW 2017, 2018
*Note	✓ 개발 환경 지원사항은 WMX3 버전에 따라 상이 할 수 있습니다. 자세한 사항은 1.7 라이브러리 항목을 참고하시기 바랍니다. ✓ C++ 과 .NET의 언어 사양으로 인해 약간의 차이가 있습니다. 1.7.3.4. C++ / .NET 라이브러리 비교 페이지에 설명되어 있습니다. 해당 페이지를 참조하시기 바랍니다.

1.7.1.3. C++ Builder

C++ 라이브러리의 모든 기능은 C++ Builder 라이브러리에서 사용할 수 있습니다.

	C++ Builder Library
라이브러리 파일	✓ C++ Builder 라이브러리는 설치 디렉토리의 Lib - C++ Builder 폴더에 설치됩니다. ✓ C++ Builder 라이브러리 파일 목록은 1.7.2. 라이브러리 목록 페이지를 참조하세요.
x86 라이브러리	WMX3 64-bit 설치에는 C++ Builder 32-bit 와 C++ Builder 64-bit 라이브러리가 모두 포함되어 있습니다
사용 방법	C++ Builder 학습서가 없습니다. IDE 의 차이로 인해 일부 구성 단계가 다를 수 있다는 것을 제외하고 학습서 1 : C++ 프로젝트 만들기의 C++ 학습서를 참조하면 됩니다.
통합 개발 환경	C++ Builder XE7 이상
****Note***	✓ 개발 환경 지원사항은 WMX3 버전에 따라 상이 할 수 있습니다. 자세한 사항은 1.7 라이브러리 항목을 참고하시기 바랍니다.

1.7.1.4. Python

C++ 모듈 라이브러리의 거의 모든 함수는 Python 라이브러리에서 사용할 수 있습니다. C++ 플랫폼 라이브러리의 함수는 Python 라이브러리에서 사용할 수 없습니다.

	Python Library
라이브러리 파일	✓ Python 라이브러리는 설치 디렉토리의 Lib - PythonApi 폴더에 설치됩니다. ✓ Python 라이브러리 파일 목록은 1.7.2. 라이브러리 목록 페이지를 참조하세요.
사용 방법	✓ Python 라이브러리 사용 방법에 대한 학습서는 학습서 5 : Python 스크립트 파일 만들기 페이지를 참조하십시오. ✓ Python 라이브러리를 사용하는 샘플 프로젝트 목록은 Python 샘플 프로젝트를 참조하십시오.
통합 개발 환경	모든 Python

1.7.1.5. RTX 응용 프로그램

C++ 라이브러리는 RTX SDK 를 사용하여 RTX 용으로 작성된 응용 프로그램에 연결될 수 있습니다. 그러나 IMDLL_RT.lib 는 IMDLL.lib 대신 링크되어야 합니다. IMDll.dll 은 RTX 응용 프로그램을 실행할 필요가 없습니다. 자세한 내용은 학습서 6: RTX SDK 를 사용하는 응용 프로그램 페이지를 참조하십시오.

1.7.2. 라이브러리 목록

아래 표는 WMX3 C++, 추가 라이브러리 요약 목록입니다.

			Windows Library		RTX Library
	Library	*Header File*	*Static Library*	*Dynamic Library*	*Static Library*
C++ Library List	IMDll	IMDef.h, IMDll.h	IMDll.lib	IMDll.dll	IMDLL_RT.lib
	IMLib	IMDef.h, IMLib.h	IMLib.lib	-	IMLib.lib
	WMX3Api	WMX3Api.h	WMX3Api.lib	-	WMX3Api.lib
	CoreMotion	CoreMotionApi.h	CoreMotionApi.lib	-	CoreMotionApi.lib
	Log	LogApi.h	LogApi.lib	-	LogApi.lib
	ApiBuffer	ApiBufferApi.h	ApiBufferApi.lib	-	ApiBufferApi.lib
	CyclicBuffer	CyclicBufferApi.h	CyclicBufferApi.lib	-	CyclicBufferApi.lib
	Compensation	CompensationApi.h	CompensationApi.lib	-	CompensationApi.lib
Direction →	IO	IOApi.h	IOApi.lib	-	IOApi.lib
	Event	EventApi.h	EventApi.lib	-	EventApi.lib
	AdvancedMotion	AdvancedMotionApi.h	AdvancedMotionApi.lib	-	AdvancedMotionApi.lib
	UserMemory	UserMemoryApi.h	UserMemoryApi.lib	-	UserMemoryApi.lib
	PMMotion	PMMotionApi.h	PMMotionApi.lib	-	PMMotionApi.lib
	EcPlatform	EcApi.h	EcApi.lib, EcApi_Win.lib	-	EcApi.lib
	RtexPlatform	RtexApi.h	RtexApi.lib	-	RtexApi.lib
	MIIIPlatform	MIIIApi.h	MIIIApi.lib	-	MIIIApi.lib
	SimuPlatform	SimuApi.h	SimuApi.lib	-	SimuApi.lib

		Windows Library			
	Library	*.Net Library*	*C++ Builder (x86)*	*C++ Builder (x64)*	*Python Library*
추가 Library List	IMDll	-	IMDll.lib	IMDll.a	-
	WMX3Api	WMX3Api_CLRLib.dll	WMX3Api.lib	WMX3Api.a	
	CoreMotion	CoreMotionApi_CLRLib.dll	CoreMotionApi.lib	CoreMotionApi.a	
	Log	LogApi_CLRLib.dll	LogApi.lib	LogApi.a	
	ApiBuffer	ApiBufferApi_CLRLib.dll	ApiBufferApi.lib	ApiBufferApi.a	
	CyclicBuffer	CyclicBufferApi_CLRLib.dll	CyclicBufferApi.lib	CyclicBufferApi.a	_WMX3ApiPython.pyd
	Compensation	CompensationApi_CLRLib.dll	CompensationApi.lib	CompensationApi.a	WMX3ApiPython.py
	IO	IOApi_CLRLib.dll	IOApi.lib	IOApi.a	
Direction →	Event	EventApi_CLRLib.dll	EventApi.lib	EventApi.a	
	AdvancedMotion	AdvancedMotionApi_CLRLib.dll	AdvancedMotionApi.lib	AdvancedMotionApi.a	
	UserMemory	UserMemoryApi_CLRLib.dll	UserMemoryApi.lib	UserMemoryApi.a	
	PMMotion	PMMotionApi_CLRLib.dll	-	-	-
	EcPlatform	EcApi_CLRLib.dll	EcApi.lib	EcApi.a	
	RtexPlatform	RtexApi_CLRLib.dll	RtexApi.lib	RtexApi.a	Not Supported
	MIIIPlatform	MIIIApi_CLRLib.dll	MIIIApi.lib	MIIIApi.a	
	SimuPlatform	SimuApi_CLRLib.dll	SimuApi.lib	SimuApi.a	

1.7.3. 라이브러리 정보

WMX3 라이브러리를 사용하기 위한 일반적인 정보를 설명합니다.

1.7.3.1. 클래스 및 구조의 초기화

WMX3 라이브러리 클래스 및 구조의 모든 멤버 변수는 정의되었을 때 0 으로 초기화 됩니다. (값은 0 으로 초기화)

1.7.3.2. Windows x64 환경에서 WMX3 x86 개발

Windows 64 비트 버전을 사용하여 WMX3 32 비트 사용자 응용 프로그램을 개발할 수 있습니다. WMX3 32 비트 설치 프로그램을 사용하여 Windows 64 비트 버전에 파일을 설치할 수 없으므로 라이브러리 파일을 수동으로 Windows 64 비트 개발 PC 에 복사해야 합니다. WMX3 32 비트 엔진에는 32 비트 버전의 RTX 가 필요하며 Windows 64 비트에서는 실행되지 않습니다. 따라서 Windows 64 비트 버전에서 개발할 때는 WMX3 32 비트 엔진을 실행해야 디버깅 기능을 사용할 수 없습니다.

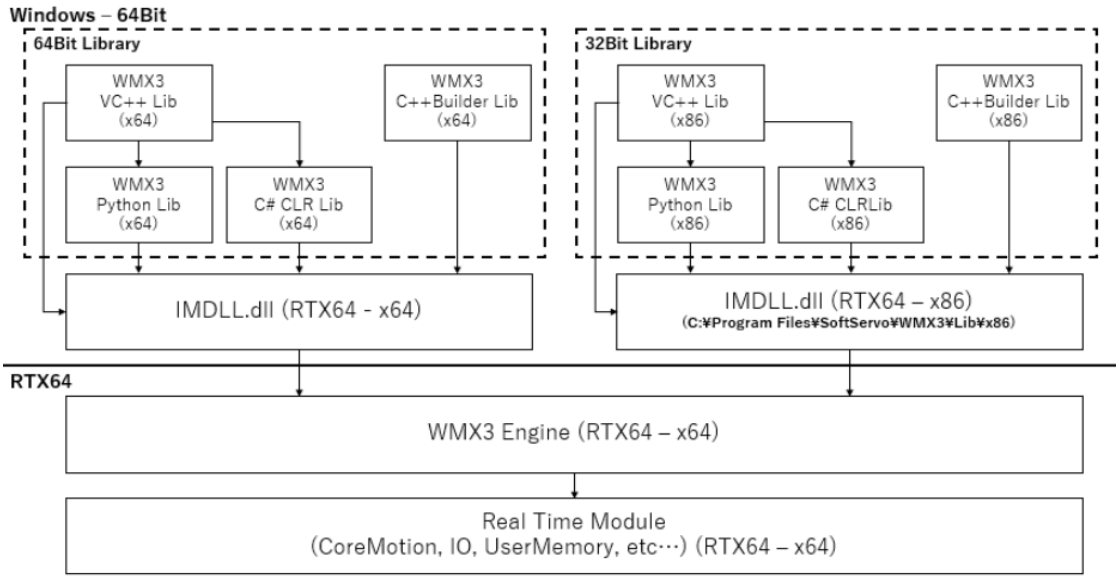
1.7.3.3. WMX3 x64 → x86 라이브러리

- WMX3 설치 폴더에는 x64, x86 응용 프로그램을 컴파일 하기 위한 두 가지 C++ 및 .NET 라이브러리가 포함되어 있습니다.
- x64 라이브러리는 [C:₩Program Files₩SoftServo₩WMX3₩Lib] 경로에 설치되며 x64 응용 프로그램을 컴파일하는데 사용됩니다. 대부분의 사용자는 이 라이브러리 세트를 사용해야 합니다.
- IMDll.dll 의 버전은 사용자 응용 프로그램을 컴파일하는데 사용된 라이브러리의 버전과 일치해야 합니다. x86 라이브러리를 사용하여 빌드 된 응용 프로그램을 실행할 때 응용 프로그램 경로의 IMDll.dll 은 Lib x86 폴더에 있어야 합니다.

- 위 사항으로 라이브러리를 사용하여 구축된 x64 와 x86 응용 프로그램은 모두 64-bit WMX3 엔진에서 실행됩니다. x64 와 x86 사용자 응용 프로그램을 동시에 실행할 수 있습니다.
- 위 사항으로 라이브러리를 사용하여 구축된 x64 와 x86 응용 프로그램은 64-bit Windows 64 비트 RTX 만 호환됩니다. 32 비트 운영 체제에서 실행될 응용 프로그램을 빌드하려면 WMX3 32-bit 가 필요합니다.

WMX3 64-bit 의 x64 와 x86 라이브러리 관계 요약도

Library



1.7.3.4. C++ / .NET 라이브러리 비교

WMX3 라이브러리는 C++ 라이브러리와 .NET 라이브러리의 두 가지 주요 패키지로 제공됩니다. 모든 WMX3 함수는 이러한 라이브러리 중 하나를 통해 실행될 수 있으며 대부분의 API 함수는 두 패키지에서 이름과 인터페이스가 동일합니다. 그러나 두 라이브러리 간에 인터페이스가 다른 영역이 몇 가지 있습니다. 이 API 참조 설명서는 C++ 인터페이스를 사용하는 대부분의 기능을 설명합니다. .NET 라이브러리 언어 (C#, VB.NET 및 C++/CLI) 사용자는 아래과 같은 차이점을 알고 있어야 합니다.

char, wchar_t* vs String*

WMX3 라이브러리의 여러 함수에는 문자열 매개변수가 있습니다. C++ 라이브러리에서 이 문자열 매개변수는 char * 또는 wchar_t* 유형 (null 로 끝나는 char 또는 wchar_t 유형의 배열)을 가지며 char *와 wchar_t* 각각에 대해 하나씩 두 개의 함수가 정의됩니다. .NET 라이브러리에서 이러한 문자열 매개변수는 문자열 유형을 가지며 하나의 함수만 정의됩니다.

C++ Library

```
SetDeviceName(char* name);
SetDeviceName(wchar_t* name);
```

.NET Library

```
SetDeviceName(String^ name);
```

기본 매개변수

C++ 라이브러리의 여러 함수에는 매개변수를 선택적으로 지정할 수 있는 기본 매개변수가 있습니다. .NET 라이브러리는 매개변수가 있는 함수 없는 함수를 별도의 함수로 정의합니다. 매개변수가 없는 함수는 C++ 라이브러리와 동일한 매개변수의 기본값을 사용합니다.

C++ Library

```
CreateDevice(char* path, DeviceType::T type, unsigned int waitTimeMilliseconds = 0, int core = -1, DWORD_PTR affinityMask = 0);
```

.NET Library

```
CreateDevice(String ^pPath, DeviceType type);
CreateDevice(String ^pPath, DeviceType type, unsigned int waitTimeMilliseconds);
CreateDevice(String ^pPath, DeviceType type, unsigned int waitTimeMilliseconds, int core);
CreateDevice(String ^pPath, DeviceType type, unsigned int waitTimeMilliseconds, int core, DWORD_PTR affinityMask);
```

unions vs child classes

C++ 라이브러리의 일부 클래스에는 동일한 클래스가 여러 데이터 세트 중 하나를 포함할 수 있는 공용체 멤버가 있습니다. 대신 .NET 라이브러리는 각 공용체 멤버에 대해 서로 다른 클래스를 정의하고 공용체와 유사한 구문을 사용하여 이러한 클래스에 액세스 할 수 있도록 속성 인터페이스를 제공합니다. 공용체 유형을 대처하는 이러한 클래스는 처음 액세스 할 때만 인스턴스화 되므로 공용체와 같이 큰 메모리 요구 사항이 없습니다.

C++ Library

```
TriggerEvent trig;

//input is an union of type
TriggerEventInputFunctionArguments
trig.input.ioBit.byteAddress = 10;
trig.input.ioBit.bitAddress = 2;
trig.inputFunction = TriggerEventInputFunction::IOBit;
```

.NET Library

```
TriggerEvent trig = new TriggerEvent();

//Input_IOBit is a property for accessing the class
trig.Input_IOBit.ByteAddress = 10;
trig.Input_IOBit.BitAddress = 2;
trig.InputFunction = TriggerEventInputFunction.IOBit;
```

배열 크기

C++ 라이브러리의 일부 함수에는 데이터로 채워질 배열을 갖춘 매개 변수와 해당 배열의 크기를 지정하는 다른 매개변수가 있습니다. 배열 객체 자체에는 배열의 크기에 대한 정보가 포함되므로 .NET 라이브러리에는 배열 매개변수만 있습니다.

C++ Library

```
UploadSDO(int slaveId, int index, int subindex, int sdoBuffSize, unsigned char sdoBuff, unsigned int actualSize, unsigned int* errCode, unsigned int waitTime);
```

.NET Library

```
UploadSDO(int index, int subindex, array<byte> ^sdoBuff, unsigned int %actualSize, unsigned int %errCode, unsigned int waitTime);
```

ErrorToString

ErrorToString 함수는 오류 코드의 문자열 표현을 반환합니다. C++ 라이브러리 ErrorToString 은 함수의 매개변수를 통해 문자열을 반환하는 반면 .NET 라이브러리 ErrorToString 은 함수 반환 값으로 String 개체를 반환합니다.

C++ Library

```
static long ErrorToString(int errCode, char *pString, unsigned int size);
static long ErrorToString(int errCode, wchar_t *pString, unsigned int size);
```

.NET Library

```
static String^ ErrorToString(int errCode);
```

1.7.4. 라이브러리 주의 사항

WMX3 라이브러리를 사용하기 전에 읽어야 할 중요한 주의 사항에 대하여 설명합니다.

1.7.4.1. RTX SDK

- 기본 WMX3 라이브러리를 사용하여 Windows 에서 실행되는 응용 프로그램을 작성할 수 있습니다. 그러나 Windows 는 실시간 운영 체제가 아니므로 API 함수 호출의 실행 타이밍에 지연 또는 데이터 값에 약간의 노이즈가 있을 수 있습니다.
- WMX3 라이브러리에는 Windows 응용 프로그램 (ApiBuffer, Trigger Motion, Events 와 같은 Api 기능)에서 지연 및 데이터 값에 약간의 노이즈를 피하는 기능이 포함되어 있지만 RTX SDK 를 사용하여 RTX 운영 체제에서 직접 실행되는 응용 프로그램을 빌드하는 방법도 있습니다.
- 일반적으로 별도의 그래픽 사용자 인터페이스 응용 프로그램이 Windows 운영 체제에서 실행되도록 작성되었습니다. 이 그래픽 사용자 응용 프로그램은 TCP/IP 또는 공유 메모리와 같은 RTX SDK 에서 사용 가능한 기능을 사용하여 RTX 응용 프로그램과 직접 통신하거나 WMX3 API 함수 호출을 사용하여 WMX3 엔진을 통해 통신을 할 수 있습니다.

- RTX SDK 를 사용하려면 별도의 IntervalZero RTX SDK 라이센스가 필요합니다. RTX SDK 를 사용하여 작성되고 RTX 운영 체제에서 실행되는 응용 프로그램은 RTX 서버 콘솔 메시지를 제외한 Windows 운영 체제의 어떤 것도 표시할 수 없습니다.

1.7.4.2. Visual Studio 2015 이상 사용자 알림

WMX3 설치에 포함된 C++ 라이브러리는 Microsoft Visual Studio 2012 용으로 컴파일 되었습니다. 아래 라이브러리는 Microsoft Visual Studio 2015 이상과 호환되지만 Microsoft 라이브러리도 연결해야 합니다.

Library

```
legacy_stdio_defibitions.lib
legacy_stdio_wide_specifiers.lib
```

이러한 라이브러리를 연결하지 않으면 liner 오류가 발생할 수 있으며 라이브러리를 연결하려면 [프로젝트 메뉴] → [속성] → [구성] → [속성] → [링커] → [입력] 이동하여 추가 족송성 옆의 세미콜론으로 구분된 목록에 라이브러리를 입력하세요.

1.7.4.3. Windows 32-bit 디버거 사용 시 경고 사항

아키텍처가 호환되지 않기 때문에 Visual Studio 디버거를 사용하여 32-bit Windows 운영체제에서 WMX 라이브러리를 사용하는 사용자 응용 프로그램을 디버깅할 때 CoreMotion, Log, IO, UserMemory 와 CyclicBuffer 클래스를 반복해서 작성 및 삭제해서는 안 되며 위의 조건에서 이러한 클래스를 작성 및 삭제하면 핸들 누수가 발생하고, 약 65536 개의 핸들이 작성되면 시스템 예외가 발생합니다. 이 문제는 32-bit 버전의 RTX 운영체제에서 32-bit Windows 를 사용할 때만 발생합니다. 64-bit Windows 운영 체제 또는 RTX64 실시간 운영 체제와 함께 WMX 라이브러리를 사용할 때는 이 문제를 고려할 필요가 없습니다.

1.8. API

WMX3 API 에 대하여 설명합니다. 각 API 의 목록과 기능에 대하여 기술되어 있으며 상세한 사용 방법 및 Sample Code 작성 예제는 본 문서를 참고하시어 WMX3 유틸리티 "API Sample Code Guide" 프로그램을 이용하시기 바랍니다. 아래의 링크를 통하여 해당 프로그램 실행하실 수 있습니다.

*Note	✓	아래 링크는 현재 PC에 WMX3가 설치된 상태에서만 연결 가능합니다.

	WMX3 API Sample Code Guide Program Start Link
Sample Code	Sample Code Guide Program.exe
Usar Manual	WMX3 Usar Manual.chm

1.8.1. API Class 목록

wmx3Api, ecApi, simuApi Namespace 내부 클래스 항목을 설명합니다.

	Class 1	Class 2	Description
wmx3Api	*WMX3Api*		디바이스, 엔진, 통신 함수가 포함된 클래스
	CoreMotion	AxisControl	축 제어 함수가 포함된 클래스
		Config	축 설정 함수가 포함된 클래스
		Home	원점 복귀 함수가 포함된 클래스
		Motion	위치 제어 함수가 포함된 클래스
		Sync	동기 제어 함수가 포함된 클래스
		Velocity	속도 제어 명령 함수가 포함된 클래스
		Torque	토크 제어 명령 함수가 포함된 클래스
	EventControl		이벤트 제어 함수가 포함된 클래스
	APIBuffer		시퀀스 버퍼 함수가 포함된 클래스
	CyclicBuffer		싸이클릭 버퍼 함수가 포함된 클래스
	Io		I/O 함수가 포함된 클래스
	Log		로그 함수가 포함된 클래스
	UserMemory		사용자 메모리 함수가 포함된 클래스
	Compensation		위치 편차 보상 관련 함수가 포함된 클래스
	AdvancedMotion	AdvMotion	고급 위치 제어 함수가 포함된 클래스
		AdvSync	고급 동기 제어 함수가 포함된 클래스
		AdvVelocity	고급 속도 제어 함수가 포함된 클래스

1.8.2. WMX3Api Class 일람

WMX3 디바이스 관리 및 기타 코어 명령이 포함된 WMX3Api 클래스 함수에 대하여 설명합니다.

	Functions	Description
WMX3Api	ErrorToString	지정된 오류 코드의 문자열 가져오기
	ApiLogToString	API 로그 데이터 문자열 가져오기
	GetLibVersion	모듈 라이브러리(버전) 가져오기
	GetIMDllVersion	IMDll 버전 가져오기
	PrintToFile	파일에 메시지 추가
	PrintToFileIfExist	파일에 메시지 추가
	GetStdOutStr	WMX3 엔진 메시지 출력 버퍼에 저장된 문자열 가져오기
	GetStdOutDataSize	WMX3 엔진 메시지 출력 버퍼에 저장된 문자열 크기 가져오기
	ClearStdOut	WMX3 엔진 메시지 출력 버퍼에 저장된 문자열 지우기
	ReleaseStdOut	WMX3 엔진 메시지 출력 버퍼와 관련된 리소스 해제

StartEngine	디바이스를 생성하지 않고 WMX3 엔진 시작
RestartEngine	현재 실행 중인 WMX3 엔진을 중지시키고 WMX3 엔진 재시작
StopEngine	현재 실행 중인 WMX3 엔진 중지
CreateDevice	WMX3 엔진과 인터페이스 연결을 위한 디바이스 생성
CloseDevice	디바이스 닫음
GetDeviceID	디바이스 ID 번호 가져오기
AutoQuitWithoutActiveDev	마지막 디바이스가 닫힐 때 엔진 자동 종료 설정
SetWatchdog	워치독 타임아웃 시간 설정
GetWatchdog	워치독 타임아웃 시간과 현재 워치독 카운트 가져오기
ResetWatchdogTimer	디바이스의 워치독 카운트 재설정
SetStatistic	API 실행 통계 데이터를 수집하도록 디바이스 구성
GetStatistic	현재 수집된 API 실행 통계 데이터 가져오기
SetInterruptId	호출 디바이스와 관련된 인터럽트 설정
GetInterruptId	호출 디바이스와 관련된 인터럽트 가져오기
StartCommunication	서보 네트워크와 통신 시작
StopCommunication	서보 네트워크와 통신 중지
GetModulesInfo	엔진에 로드된 모듈 정보 가져오기
GetModuleInfo	엔진에 로드된 하나의 모듈 정보 가져오기
GetEngineStatus	엔진 상태 가져오기
GetAllDevices	모든 기존 디바이스 정보 가져오기
SetDeviceName	디바이스 이름 설정
GetDeviceName	디바이스 이름 가져오기
SetModuleSuspend	통신 시작 시 특정 플랫폼 모듈을 로드 중지할 지 설정
GetModuleSuspend	통신 시작 시 특정 플랫폼 모듈 로드 중지 여부에 대한 설정 가져오기
AbortModuleSuspend	통신 시작 시 특정 플랫폼 모듈 로드 일시 중지 여부에 대한 모든 설정 삭제
SleepAtEngine	지정된 타임아웃 간격이 경과할 때까지 현재 스레드 실행을 일시 중지
PrintToServerConsole	RTX Server Console 에 메시지를 표시
RecordWindowsUpdates	설치된 Windows Update 목록을 파일에 기록
CompareWindowsUpdates	현재 설치된 Windows Update 목록을 파일에 기록된 목록과 비교

1.8.3. CoreMotion :: AxisControl Class 일람

CoreMotion 클래스 내 축 제어 명령이 포함된 AxisControl 클래스 함수에 대하여 설명합니다.

	Functions	*Description*
AxisControl	SetServoOn	서보 네트워크에서 서보 드라이브 On 또는 Off 설정
	ClearAmpAlarm	지정된 축에 대한 서보 드라이브 앰프 알람 해제
	ClearAxisAlarm	지정된 축에 대한 축 알람 해제
	SetAxisCommandMode	축에 대한 명령 모드 설정
	GetAxisCommandMode	축의 명령 모드 가져오기
	GetPosCommand	축의 현재 사이클 지령 위치 가져오기
	GetPosFeedback	축의 현재 사이클 피드백 위치 가져오기
	GetVelCommand	축의 현재 사이클 지령 속도 가져오기
	GetVelFeedback	축의 현재 사이클 피드백 속도 가져오기

1.8.4. CoreMotion :: Config Class 일람

CoreMotion 클래스 내 축 설정 명령이 포함된 Config 클래스 함수에 대하여 설명합니다.

	Functions	*Description*
Config	SetParam	모든 축에 대한 시스템 매개 변수 설정
	GetParam	모든 축에 대한 시스템 매개 변수 설정 가져오기
	SetAxisParam	모든 축에 대한 축 매개 변수 설정
	GetAxisParam	모든 축에 대한 축 매개 변수 설정 가져오기
	SetGearRatio	축 기어비 설정
	SetSingleTurn	축의 싱글턴 모드 및 싱글턴 엔코더 카운터 설정
	SetMovingAverageProfileTime	축에 대한 이동 평균 프로파일 시간 매개 변수 설정
	SetAxisUnit	축에 대한 축 최소 단위 매개 변수 설정
	SetVelocityFeedforwardGain	축에 대한 속도 피드 포워드 게인 매개 변수 설정
	SetAxisPolarity	축 방향 설정
	SetAbsoluteEncoderMode	축에 대한 앱솔루트 엔코더 모드 설정
	SetAbsoluteEncoderHomeOffset	축에 대한 앱솔루트 엔코더 원점 오프셋 설정
	GetGearRatio	축의 기어비 가져오기

GetSingleTurn	축의 싱글턴 모드 및 싱글턴 엔코더 카운터 가져오기
GetMovingAVerageProfileTime	축의 이동 평균 프로파일 시간 매개 변수 가져오기
GetAxisUnit	축에 대한 축 최소 단위 매개 변수 가져오기
GetVelocityFeedforwardGain	축의 속도 피드 포워드 게인 매개 변수 가져오기
GetAxisPolarity	축 극성 가져오기
GetAbsoluteEncoderMode	축의 앱솔루트 엔코더 모드 가져오기
GetAbsoluteEncoderHomeOffset	축의 앱솔루트 엔코더 원점 오프셋 가져오기
SetFeedbackParam	축의 피드백 매개 변수 설정
SetHomeParam	축의 원점 매개 변수 설정
SetLimitParam	축의 리밋 매개 변수 설정
SetMotionParam	축의 모션 매개 변수 설정
SetAlarmParam	축의 알람 매개 변수 설정
SetSyncParam	축의 싱크 매개 변수 설정
SetFlightRecorderParam	Flight Recorder 매개 변수 설정
SetFlightRecorderPath	Flight Recorder Data 를 저장할 파일 경로 설정
SetEmergencyStopParam	비상 정지 매개 변수 설정
GetFeedbackParam	축의 피드백 매개 변수 가져오기
GetHomeParam	축의 원점 매개 변수 가져오기
GetLimitParam	축의 리밋 매개 변수 가져오기
GetMotionParam	축의 모션 매개 변수 가져오기
GetAlarmParam	축의 알람 매개 변수 가져오기
GetSyncParam	축의 싱크 매개 변수 가져오기
GetFlightRecorderParam	Flight Recorder 매개 변수 설정 가져오기
GetEmergencyStopParam	비상 정지 매개 변수 가져오기
GetDefaultParam	기본 시스템 매개 변수 가져오기
GetDefaultAxisParam	기본 축 매개 변수 가져오기
Export	시스템 매개 변수를 xml 파일로 내보내기
Import	xml 파일에서 시스템 매개 변수 가져오기
GetAndExportAll	모든 축의 현재 매개 변수를 xml 파일로 내보내기
ImportAndSetAll	xml 파일의 매개 변수를 모든 축의 현재 매개 변수로 설정

1.8.5. CoreMotion :: Home Class 일람

CoreMotion 클래스 내 원점 복귀 명령이 포함된 Home 클래스 함수에 대하여 설명합니다.

	Functions	*Description*
Home	StartHome	축 원점 복귀 시작
	Continue	원점 복귀 모션 중 일시 정지된 축을 계속 원점 복귀
	Cancel	원점 복귀 모션 중 일시 정지된 축에 대한 원점 복귀 취소
	SetCommandPos	축의 현재 지령 위치를 지정된 값으로 설정
	SetFeedbackPos	축의 현재 피드백 위치를 지정된 값으로 설정
	SetHomeDone	원점 완료 상태를 설정
	SetCommandPosToFeedbackPos	현재 지령 위치와 피드백 위치를 동일하게 설정
	GetHomeData	모든 축에 대한 원점 복귀 관련 데이터 가져오기

1.8.6. CoreMotion :: Motion Class 일람

CoreMotion 클래스 내 위치 명령이 포함된 Motion 클래스 함수에 대하여 설명합니다.

	Functions	*Description*
Motion	StartPos	축에 대한 절대 위치 모션 명령을 시작
	StartMov	축에 대한 상대 위치 모션 명령을 시작
	StartLinearIntplPos	절대 위치 선형 보간 모션 명령 시작
	StartLinearIntplMov	상대 위치 선형 보간 모션 명령 시작
	StartCircularIntplPos	원호 보간 모션 명령 시작
	StartCircularIntplMov	원호 보간 모션 명령 시작
	StartHelicalIntplPos	헬리컬 보간 모션 명령 시작
	StartHelicalIntplMov	헬리컬 보간 모션 명령 시작
	StartVelToPos	현재 속도 명령을 실행 중인 축에 대해 절대 트리거 위치 명령 시작
	StartVelToMov	현재 속도 명령을 실행 중인 축에 대해 상대 트리거 위치 명령 시작
	StartTrqToPos	현재 토크 명령을 실행 중인 축에 대해 절대 트리거 위치 명령 시작
	StartTrqToMov	현재 토크 명령을 실행 중인 축에 대해 상대 트리거 위치 명령 시작
	StartJog	축에 대한 조그 모션 명령 시작

StartPosToJog	현재 위치 명령을 실행 중인 축에 대해 트리거된 조그 명령 시작
StartMovToJog	상대 위치 명령과 트리거된 조그 명령 동시에 시작
Stop	축의 모션 정지
ExecQuickStop	Quick Stop Dec 매개 변수를 사용하여 축의 모션 정지
ExecTimedStop	사다리꼴 프로파일을 사용하여 지정된 시간 동안 현재 모션 중인 축 정지
Wait	차단 대기 명령을 시작
Pause	축에 대한 위치 명령 또는 보간 명령 실행 일시 정지
Resume	일시 정지된 위치 명령 또는 축에 대한 보간 명령 다시 실행
OverridePos	현재 위치 명령을 실행 중인 축의 목표 위치를 무시
OverrideMov	현재 위치 명령을 실행 중인 축의 목표 위치 무시
OverrideVel	현재 위치, 조그, 속도 명령을 실행 중인 축의 속도 무시
OverrideAcc	현재 위치, 조그, 속도 명령을 실행 중인 축의 가속 무시
OverrideDec	현재 위치, 조그, 속도 명령을 실행 중인 축의 감속 무시
OverrideJerkAcc	현재 위치, 조그, 속도 명령을 실행 중인 축의 가속 저크 무시
OverrideJerkDec	현재 위치, 조그, 속도 명령을 실행 중인 축의 감속 저크 무시
OverrideProfile	현재 위치, 조그, 속도 명령을 실행 중인 축의 전체 프로파일 무시
StopJogAtPos	조그 명령을 실행 중인 축을 지정한 위치에 정확하게 멈추도록 감속 정지
SuperimposeMov	중첩된 모션 명령을 상대 위치에서 시작
StopSuperimpose	중첩 모션 명령 중지
SimulatePos	실제로 축을 이동하지 않고 위치 모션 명령을 시뮬레이션
SimulateLinearIntplPos	실제로 축을 이동하지 않고 선형 보간 모션 명령을 시뮬레이션
SimulatePosAtTime	실제로 축을 이동하지 않고 위치 모션 명령을 시뮬레이션
SimulateTimeAtPos	실제로 축을 이동하지 않고 위치 모션 명령을 시뮬레이션
SimulateTimeAtDist	실제로 축을 이동하지 않고 선형 보간 모션 명령을 시뮬레이션

Note

- ✓ Simulate 시뮬레이션 API의 경우 모션 명령, 프로파일 등 시간이 반환됩니다.
- ✓ StartCircularIntplPos 또는 StartCircularIntplMov Circular Interpolation 명령은 조건에 따라 여러 가지 형태로 나뉘어져 있습니다.

1.8.7. CoreMotion :: Sync Class 일람

CoreMotion 클래스 내 동기 제어 명령이 포함된 Sync 클래스 함수에 대하여 설명합니다.

	Functions	*Description*
Sync	SetSyncMasterSlave	마스터 축과 슬레이브 축 간의 동기 제어 설정
	SetSyncCombine	두 마스터 축의 위치 명령을 결합하여 슬레이브 축의 위치 명령으로 설정
	SetAbsoluteSyncPhase	슬레이브 축의 절대 동기화 위상 설정
	AddRelativeSyncPhase	슬레이브 축에 상대 동기화 위상 추가
	SetSyncGearRatio	동기 슬레이브 축과 마스터 축 간의 동기 기어비 설정
	SyncToJog	슬레이브 축에 대한 동기화를 해제하고 즉시 슬레이브 축에 대한 조그 명령 시작
	ResolveSync	동기 제어에서 지정된 슬레이브 축 해제
	SetSyncGroup	동기화 그룹의 축 및 매개 변수 설정
	GetSyncGroup	동기화 그룹의 축 및 매개 변수 가져오기
	AddAxisToSyncGroup	기존 동기화 그룹에 다른 축 추가
	RemoveAxisFromSyncGroup	기존 동기화 그룹에서 축 제거
	EnableSyncGroup	동기화 그룹 활성화 또는 비활성화
	GetSyncGroupStatus	지정된 동기화 그룹의 현재 상태 읽기
	ClearSyncGroupError	지정된 동기화 그룹의 모든 동기화 그룹 오류 지우기

1.8.8. CoreMotion :: Velocity Class 일람

CoreMotion 클래스 내 속도 제어 명령이 포함된 Velocity 클래스 함수에 대하여 설명합니다.

	Functions	*Description*
Velocity	StartVel	축에 대한 속도 명령 시작
	Stop	속도 명령을 실행하는 축의 모션 중지
	ExecQuickStop	속도 지령을 실행하고 있는 축의 모션을 Quick Stop 으로 정지
	SetMaxMotorSpeed	축의 최대 모터 속도 설정
	GetMaxMotorSpeed	축의 최대 모터 속도 가져오기
	OverrideVel	현재 속도 명령을 실행 중인 축에 대한 오버라이드 속도 설정
	StartPosToVel	현재 위치 명령을 실행 중인 축에 대해 트리거된 속도 명령 시작
	StartPosToVel	현재 위치 명령을 실행 중인 축에 대해 트리거된 속도 명령 시작

1.8.9. CoreMotion :: Torque Class 일람

CoreMotion 클래스 내 토크 제어 명령이 포함된 Torque 클래스 함수에 대하여 설명합니다.

	Functions	*Description*
Torque	SetMaxTrqLimit	축의 최대 토크 리미트 설정
	GetMaxTrqLimit	축의 최대 토크 리미트 가져오기
	SetPositiveTrqLimit	축의 정방향의 최대 토크 리미트 설정
	GetPositiveTrqLimit	축의 정방향에서 최대 토크 리미트 가져오기
	SetNegativeTrqLimit	축의 역방향의 최대 토크 리미트 설정
	GetNegativeTrqLimit	축의 역방향에서 최대 토크 리미트 가져오기
	StartTrq	축에 대한 토크 모션 명령 시작
	StartRampTimeTrq	축에 대해 시간에 따라 토크가 변경되는 토크 모션 명령 시작
	StopTrq	축에 대한 토크 정지 명령
	StartPosToTrq	현재 위치 명령을 실행 중인 축에 대해 트리거된 토크 명령 시작
	StartVelToTrq	현재 속도 명령을 실행 중인 축에 대해 트리거된 토크 명령 시작

1.8.10. EventControl Class 일람

EventControl 클래스 함수에 대하여 설명합니다.

	Functions	*Description*
EventControl	GetVersion	모듈 버전 정보 가져오기
	SetEvent	이벤트 설정
	GetEventModuleId	기존 이벤트의 입력 함수 모듈 ID 및 출력 함수 모듈 ID 가져오기
	GetEvent	이벤트 매개 변수 가져오기
	GetEventOption	기존 이벤트의 옵션 설정 가져오기
	SetEventInput	이벤트의 입력 함수 설정
	SetEventOutput	이벤트의 출력 함수 설정
	GetEventInput	이벤트의 입력 함수 가져오기
	GetEventOutput	이벤트의 출력 함수 가져오기
	GetAllEventID	모든 기존 이벤트의 ID 가져오기
	EnableEvent	기존 이벤트 활성화 또는 비활성화
	RemoveEvent	기존 이벤트 제거
	ClearAllEvent	모든 기존 이벤트 제거
	SetSoftwareTouchProbe	소프트웨어 터치 프로브 채널에 대한 매개 변수 설정
	EnableSoftwareTouchProbe	소프트웨어 터치 프로브 채널을 활성화
	GetSoftwareTouchProbe	소프트웨어 터치 프로브 채널의 매개 변수 가져오기
	IsSoftwareTouchProbeLatched	소프트웨어 터치 프로브 채널에 래치된 데이터가 있는지 확인
	GetSoftwareTouchProbeCounterValue	소프트웨어 터치 프로브의 래치 데이터 가져오기
	SetHardwareTouchProbe	축의 하드웨어 터치 프로브에 대한 매개 변수 설정
	GetHardwareTouchProbeStatus	축의 하드웨어 터치 프로브의 매개 변수 및 현재 상태 가져오기
	EnableHardwareTouchProbe	축의 하드웨어 터치 프로브를 활성화 또는 비활성화
	SetPSOConfig	위치 동기 출력 채널에 대한 매개 변수 설정
	GetPSOConfig	위치 동기 출력 채널에 대한 매개 변수 가져오기
	SetPSOSingleData	위치 동기 출력 채널에 대한 단일 데이터 포인트 설정
	SetPSOMultipleData	위치 동기 출력 채널에 대해 여러 데이터 포인트 설정
	SetPSOIntervalData	위치 동기 출력 채널 범위의 데이터 포인트 설정
	GetPSOData	위치 동기 출력 채널에 대해 현재 설정된 모든 데이터 포인트 가져오기
	GetPSODataCount	위치 동기 출력 채널에 대해 현재 설정된 데이터 포인트 수 가져오기
	GetPSOIntervalData	위치 동기 출력 채널의 범위 및 간격 가져오기
	StartPSO	위치 동기 출력 채널 시작
	GetPSOStatus	위치 동기 출력 채널의 상태 가져오기
	SetPlannedVelOverrideConfig	지정된 속도 오버라이드 채널에 대한 매개 변수 설정
	GetPlannedVelOverrideConfig	지정된 속도 오버라이드 채널에 대한 매개 변수 가져오기
	SetPlannedVelOverrideSingleData	지정된 속도 오버라이드 채널에 대해 단일 데이터 포인트를 설정
	SetPlannedVelOverrideMultipleData	지정된 속도 오버라이드 채널에 대해 여러 데이터 포인트를 설정
	GetPlannedVelOverrideData	지정된 속도 오버라이드 채널에 대해 현재 설정된 모든 데이터 포인트 가져오기
	GetPlannedVelOverrideDataCount	지정된 속도 오버라이드 채널에 대해 현재 설정된 데이터 포인트 수 가져오기
	StartPlannedVelOverride	지정된 속도 오버라이드 채널 시작
	GetPlannedVelOverrideStatus	지정된 속도 오버라이드 채널의 상태 가져오기

1.8.11. ApiBuffer Class 일람

ApiBuffer 클래스 함수에 대하여 설명합니다.

	Functions	Description
ApiBuffer	GetVersion	모듈 버전 정보 가져오기
	CreateApiBuffer	API 버퍼와 함께 사용할 메모리 공간 생성
	FreeApiBuffer	API 버퍼 메모리 공간 닫음
	StartRecordBufferChannel	API 버퍼 채널에 API 기록 시작
	EndRecordBufferChannel	API 버퍼 채널에 API 기록 종료
	GetRecordingBufferChannel	현재 기록 중인 API 버퍼 채널 가져오기
	Execute	API 버퍼 실행
	Halt	API 버퍼 중지
	Clear	API 버퍼 삭제
	Rewind	API 버퍼 되돌리기
	GetStatus	API 현재 상태 가져오기
	SetOptions	API 버퍼 옵션 설정
	GetOptions	API 버퍼에 현재 적용된 옵션 가져오기
	SetWatch	API 버퍼에 대한 감시 옵션 설정
	GetWatch	API 버퍼에 대한 감시 옵션 가져오기
	Sleep	API 버퍼 중지 명령 추가
	Wait	API 버퍼 대기 명령 추가
	FlowIf	API 버퍼에 If 명령 추가
	FlowElseIf	API 버퍼에 Else If 명령 추가
	FlowElse	API 버퍼에 Else 명령 추가
	FlowEndIf	API 버퍼에 End If 명령 추가

1.8.12. CyclicBuffer Class 일람

CyclicBuffer 클래스 함수에 대하여 설명합니다.

	Functions	Description
CyclicBuffer	GetVersion	모듈 버전 정보 가져오기
	OpenCyclicBuffer	축에 대한 새로운 Cyclic Buffer 메모리 공간 생성
	CloseCyclicBuffer	축의 Cyclic Buffer 메모리 공간 닫음
	AddCommand	축의 Cyclic Position 명령 버퍼에 위치 명령 데이터 추가
	AddCommand	축의 Cyclic Position 명령 버퍼에 임의의 위치 명령 데이터 추가
	AddCommand	축의 Cyclic Position 명령 버퍼에 각각의 위치 명령 데이터 추가
	AddCommand	축의 Cyclic Position 명령 버퍼에 원하는 위치 명령 데이터 추가
	Execute	축에 대한 Cyclic Position 명령 버퍼 실행 시작
	Abort	축의 Cyclic Position 명령 버퍼의 실행을 중지하고 삭제
	GetStatus	지정된 축의 Cyclic Position 명령 버퍼 상태를 가져옵니다.

*Note ✓ OpenCyclicBuffer, CloseCyclicBuffer는 다 축 설정이 가능합니다.

1.8.13. Io Class 일람

Io 클래스 함수에 대하여 설명합니다.

	Functions	Description
Io	GetVersion	모듈 버전 정보 가져오기
	SetOutBit	디지털 출력 비트 값 설정
	SetOutByte	디지털 출력 바이트 값 설정
	SetOutBytes	디지털 출력 바이트 배열 값 설정
	SetOutBits	디지털 출력 비트 배열 값 설정
	SetOutAnalogDataChar	아날로그 출력 1 바이트(Signed) 값 설정
	SetOutAnalogDataUChar	아날로그 출력 1 바이트(Unsigned) 값 설정
	SetOutAnalogDataShort	아날로그 출력 2 바이트(Signed) 값 설정
	SetOutAnalogDataUShort	아날로그 출력 2 바이트(Unsigned) 값 설정
	SetOutAnalogDataInt	아날로그 출력 4 바이트(Signed) 값 설정
	SetOutAnalogDataUInt	아날로그 출력 4 바이트(Unsigned) 값 설정

	GetInBit	디지털 입력 비트 값 가져오기
	GetInByte	디지털 입력 바이트 값 가져오기
	GetInBytes	디지털 입력 바이트 배열 값 가져오기
	GetInAnalogDataChar	아날로그 입력 1 바이트(Signed) 값 가져오기
	GetInAnalogDataUChar	아날로그 입력 1 바이트(Unsigned) 값 가져오기
	GetInAnalogDataShort	아날로그 입력 2 바이트(Signed) 값 가져오기
	GetInAnalogDataUShort	아날로그 입력 2 바이트(Unsigned) 값 가져오기
	GetInAnalogDataInt	아날로그 입력 4 바이트(Signed) 값 가져오기
	GetInAnalogDataUInt	아날로그 입력 4 바이트(Unsigned) 값 가져오기
	GetOutBit	디지털 출력 비트 값 가져오기
	GetOutByte	디지털 출력 바이트 값 가져오기
	GetOutBytes	디지털 출력 바이트 배열 값 가져오기
	GetOutAnalogDataChar	아날로그 출력 1 바이트(Signed) 값 가져오기
	GetOutAnalogDataUChar	아날로그 출력 1 바이트(Unsigned) 값 가져오기
	GetOutAnalogDataShort	아날로그 출력 2 바이트(Signed) 값 가져오기
	GetOutAnalogDataUShort	아날로그 출력 2 바이트(Unsigned) 값 가져오기
	GetOutAnalogDataInt	아날로그 출력 4 바이트(Signed) 값 가져오기
	GetOutAnalogDataUInt	아날로그 출력 4 바이트(Unsigned) 값 가져오기
	SetOutBitEx	디지털 출력 비트 값 설정
	SetOutByteEx	디지털 출력 바이트 값 설정
	SetOutBytesEx	디지털 출력 바이트 배열 값 설정
	SetOutBitsEx	디지털 출력 비트 배열 값 설정
	SetOutAnalogDataCharEx	아날로그 출력 1 바이트(Signed) 값 설정
	SetOutAnalogDataUCharEx	아날로그 출력 1 바이트(Unsigned) 값 설정
	SetOutAnalogDataShortEx	아날로그 출력 2 바이트(Signed) 값 설정
	SetOutAnalogDataUShortEx	아날로그 출력 2 바이트(Unsigned) 값 설정
	SetOutAnalogDataIntEx	아날로그 출력 4 바이트(Signed) 값 설정
	SetOutAnalogDataUIntEx	아날로그 출력 4 바이트(Unsigned) 값 설정
	GetInBitEx	디지털 입력 비트 값 가져오기
	GetInByteEx	디지털 입력 바이트 값 가져오기
	GetInBytesEx	디지털 입력 바이트 배열 값 가져오기
	GetInAnalogDataCharEx	아날로그 입력 1 바이트(Signed) 값 가져오기
	GetInAnalogDataUCharEx	아날로그 입력 1 바이트(Unsigned) 값 가져오기
	GetInAnalogDataShortEx	아날로그 입력 2 바이트(Signed) 값 가져오기
	GetInAnalogDataUShortEx	아날로그 입력 2 바이트(Unsigned) 값 가져오기
	GetInAnalogDataIntEx	아날로그 입력 4 바이트(Signed) 값 가져오기
	GetInAnalogDataUIntEx	아날로그 입력 4 바이트(Unsigned) 값 가져오기
	GetOutBitEx	디지털 출력 비트 값 가져오기
	GetOutByteEx	디지털 출력 바이트 값 가져오기
	GetOutBytesEx	디지털 출력 바이트 배열 값 가져오기
	GetOutAnalogDataCharEx	아날로그 출력 1 바이트(Signed) 값 가져오기
	GetOutAnalogDataUCharEx	아날로그 출력 1 바이트(Unsigned) 값 가져오기
	GetOutAnalogDataShortEx	아날로그 출력 2 바이트(Signed) 값 가져오기
	GetOutAnalogDataUShortEx	아날로그 출력 2 바이트(Unsigned) 값 가져오기
	GetOutAnalogDataIntEx	아날로그 출력 4 바이트(Signed) 값 가져오기
	GetOutAnalogDataUIntEx	아날로그 출력 4 바이트(Unsigned) 값 가져오기
	SetInitialOutByte	통신 시작 시 적용되는 디지털 출력 바이트의 초기 값 설정
	SetInitialOutBytes	통신 시작 시 적용되는 디지털 출력 바이트 배열의 초기 값 설정
	GetInitialOutByte	통신 시작 시 적용되는 디지털 출력 바이트의 초기 값 가져오기
	GetInitialOutBytes	통신 시작 시 적용되는 디지털 출력 바이트 배열의 초기 값 가져오기
	GetInitialOutByteInterruptId	지정된 디지털 출력 바이트의 초기 값을 설정할 인터럽트의 ID 가져오기
	GetInitialOutBytesInterruptId	지정된 디지털 출력 바이트 배열의 초기 값을 설정할 인터럽트 ID 를 가져오기

***Note** ✓ Ex 표시된 함수들은 WMX3 엔진이 인터럽트를 처리하고 있어도 딜레이 없이 즉각적인 실행이 가능한 함수입니다.

1.8.14. Log Class 일람

Log 클래스 함수에 대하여 설명합니다.

	Functions	*Description*
Log	GetVersion	모듈 버전 정보 가져오기
	StartLog	데이터 로깅 시작

StopLog	데이터 로깅 중지
ResetLog	지정된 로그 채널에 대한 로그 설정, 옵션 및 상태 재설정
SetLogHeader	로그 파일의 시작 부분에 복사되는 로그 헤더 문자열 설정
SetLog	로그 작업에서 수집할 데이터 지정
SetLogOption	로그 채널에 대한 로그 옵션 설정
GetLogOption	로그 채널에 대한 로그 옵션 가져오기
SetLogFilePath	로그 채널의 파일 경로 설정
GetLogFilePath	로그 채널의 파일 경로 가져오기
GetLogStatus	데이터 로그 작업의 현재 상태 가져오기
GetDetailLogStatus	데이터 로그 작업의 현재 상태 가져오기
OpenMemoryLogBuffer	메모리 로그 버퍼 열기
CloseMemoryLogBuffer	메모리 로그 버퍼 닫기
SetMemoryLog	메모리 로그 설정
SetMemoryIOLog	I/O 메모리 로그 설정
SetMemoryMLog	사용자 메모리 로그 설정
StartMemoryLog	메모리 로그 작업 시작
StopMemoryLog	메모리 로그 작업 중지
GetMemoryLogStatus	메모리 로그 작업의 현재 상태 가져오기
GetMemoryLogData	메모리 로그에서 데이터 검색
SetApiLog	API 로그에 대한 매개 변수 설정
StartApiLog	API 로그에 대한 로깅 시작
StopApiLog	API 로그에 대한 로깅 중지
GetApiLogStatus	API 로그에 대한 현재 상태 가져오기
OpenApiLogFile	API 로그 파일을 열어 정보 가져오기
GetApiLogData	API 로그 파일에서 API 함수 호출 정보 가져오기
CloseApiLogFile	API 로그 파일 닫기

1.8.15. UserMemory Class 일람

UserMemory 클래스 함수에 대하여 설명합니다.

	Functions	*Description*
UserMemory	GetVersion	모듈 버전 정보 가져오기
	SetMBit	사용자 메모리 디지털 비트 값 설정
	SetMByte	사용자 메모리 디지털 바이트 값 설정
	SetMBytes	사용자 메모리 디지털 바이트 배열 값 설정
	SetMBits	사용자 메모리 디지털 비트 배열 값 설정
	SetMAnalogDataChar	1 바이트의 사용자 메모리 아날로그 데이터 값 설정
	SetMAnalogDataUChar	1 바이트의 사용자 메모리 아날로그 데이터 값 설정
	SetMAnalogDataShort	2 바이트의 사용자 메모리 아날로그 데이터 값 설정
	SetMAnalogDataUShort	2 바이트의 사용자 메모리 아날로그 데이터 값 설정
	SetMAnalogDataInt	4 바이트의 사용자 메모리 아날로그 데이터 값 설정
	SetMAnalogDataUInt	4 바이트의 사용자 메모리 아날로그 데이터 값 설정
	GetMBit	사용자 메모리 디지털 비트 값 가져오기
	GetMByte	사용자 메모리 디지털 바이트 값 가져오기
	GetMBytes	사용자 메모리 디지털 바이트 배열 값 가져오기
	GetMAnalogDataChar	1 바이트의 사용자 메모리 아날로그 데이터 값 가져오기
	GetMAnalogDataUChar	1 바이트의 사용자 메모리 아날로그 데이터 값 가져오기
	GetMAnalogDataShort	2 바이트의 사용자 메모리 아날로그 데이터 값 가져오기
	GetMAnalogDataUShort	2 바이트의 사용자 메모리 아날로그 데이터 값 가져오기
	GetMAnalogDataInt	4 바이트의 사용자 메모리 아날로그 데이터 값 가져오기
	GetMAnalogDataUInt	4 바이트의 사용자 메모리 아날로그 데이터 값 가져오기
	SetMBitEx	사용자 메모리 디지털 비트 값 설정
	SetMByteEx	사용자 메모리 디지털 바이트 값 설정
	SetMBytesEx	사용자 메모리 디지털 바이트 배열 값 설정
	SetMBitsEx	사용자 메모리 디지털 비트 배열 값 설정
	SetMAnalogDataCharEx	1 바이트의 사용자 메모리 아날로그 데이터 값 설정
	SetMAnalogDataUCharEx	1 바이트의 사용자 메모리 아날로그 데이터 값 설정
	SetMAnalogDataShortEx	2 바이트의 사용자 메모리 아날로그 데이터 값 설정
	SetMAnalogDataUShortEx	2 바이트의 사용자 메모리 아날로그 데이터 값 설정
	SetMAnalogDataIntEx	4 바이트의 사용자 메모리 아날로그 데이터 값 설정
	SetMAnalogDataUIntEx	4 바이트의 사용자 메모리 아날로그 데이터 값 설정
	GetMBitEx	사용자 메모리 디지털 비트 값 가져오기

GetMByteEx	사용자 메모리 디지털 바이트 값 가져오기
GetMBytesEx	사용자 메모리 디지털 바이트 배열 값 가져오기
GetMAnalogDataCharEx	1 바이트의 사용자 메모리 아날로그 데이터 값 가져오기
GetMAnalogDataUCharEx	1 바이트의 사용자 메모리 아날로그 데이터 값 가져오기
GetMAnalogDataShortEx	2 바이트의 사용자 메모리 아날로그 데이터 값 가져오기
GetMAnalogDataUShortEx	2 바이트의 사용자 메모리 아날로그 데이터 값 가져오기
GetMAnalogDataIntEx	4 바이트의 사용자 메모리 아날로그 데이터 값 가져오기
GetMAnalogDataUIntEx	4 바이트의 사용자 메모리 아날로그 데이터 값 가져오기
GetUserMemoryAddress	사용자 메모리 주소 공간의 메모리 주소에 대한 포인터 가져오기

***Note**

✓ Ex 표시된 함수들은 WMX3 엔진이 인터럽트를 처리하고 있어도 딜레이 없이 즉각적인 실행이 가능한 함수입니다.

1.8.16. API 에러 코드

1.9. 유틸리티

1.9.1. WOS (WMX3 Operating Station)

1.9.1.1. WOS 개요

1.9.1.2. WOS 시스템

2. RTX

3. Guide