PERFECTION IN AUTOMATION
www.br-automation.com

TM213
Automation Runtime

버전 정보

버전	날짜	수정내역	번역	검수
1.0	2016.11.04	첫번째 버전 TM213TRE.40-ENG_Automation Runtime_V4100	정진강	임은

Table 1: Versions

선행 및 필요 조건

교육 자료	TM210 – Automation Studio 사용하기
소프트웨어	Automation Studio 4.0
하드웨어	X20CP1486

목차

1 소개

실시간 시스템(Real-time operating system), Automation Runtime 은 Automation Studio 의 필수 구성요소이다.
Automation Runtime 은 B&R 타겟 시스템에서 어플리케이션이 작동하도록 소프트웨어 커널을 구성한다.

Automation Runtime 은 모듈식 구성이 특징이며, 정확한 시간 안에 어플케이션을 빠르고 반복적으로 실행한다. 실행 시간 동안 품질과 정확성을 보장하면서 최적의 제품 생산 수량 충족시킨다.

그림 1 Automation Runtime 타켓 시스템

이 교육 자료는 Automation Runtime 의 일반적인 개요와 특징에 대해 설명할 것이다.

1.1 학습 목표

이 교육 자료는 Automation Studio 에서 당신의 어플리케이션 개발시 Automation Runtime 을 어떻게 설정하는지 돕기 위하여 일반적인 응용 프로그램을 예를 들어 설명한다.

- 자동화 분야에서 실시간 시스템(Real-time operating system)에 어떠한 요구사항들이 있는지 배울 것이다.
- Automation Runtime 특징과 기능에 대해 배울 것이다.
- 균일한 런타임 시스템이 통합 자동화 솔루션에 어떠한 이점이 있는지 배울 것이다.
- 메모리 관리, 변수 선언 범위, RETAIN 변수가 Automation Runtime 에서 어떻게 다뤄지는 배울 것이다.
- B&R 제어기 시작과 런타임 작동 원리에 대해 배울 것이다.
- 어떻게 Automation Runtime I/O 관리가 작동되는지 배울 것이다.
- 멀티태스킹에 관련된 상호 관계에 대해 배울 것이다.
- Automation Runtime 의 설정 옵션에 대해 배울 것이다.

2 실시간 시스템(Real-time operating system)

Automation Runtime 은 어플리케이션을 생성하기 위해서 결정성, 하드웨어 독립성, 멀티태스킹 툴 등을 사용자에게 제공한다.

Automation Runtime 에 있는 IEC 라이브러리 펑션(IEC library functions)은 오류를 방지하는데 도움을 주어 빠르고 쉽게 프로그래밍을 한다.

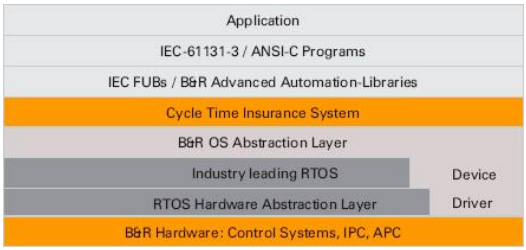


그림 2 Operating system 구성도

Operating system 은 두 가지 주요 작업을 수행한다: 사용자가 세세한 모든 부분을 신경 쓸 필요가 없도록 하드웨어/소프트웨어 리소스 관리와 정형적인 하드웨어 접속 방법을 제공.

프로그램 – 또는 테스크(task) – 에 주기적으로 실행되는 명확한 실행시간, 또는 작업 클래스 등이 할당될 수 있다.

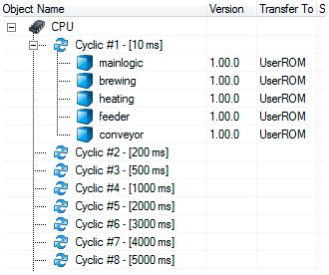


그림 3 실행중인 프로그램 시간 체계

Real-time operating system

2.1 요구사항과 기능

Automation Runtime 은 B&R 타겟 시스템에 통합되어 있다. 어플리케이션 프로그램이 I/O 시스템, 인터페이스, fieldbuses, 네트워크 그리고 저장장치 등에 접근할 수 있다.

그림 4: Automation Runtime 의 특성들

Automation Runtime 은 중요한 기능을 제공한다:

- 모든 B&R 타겟 시스템에서 작동한다.
- 어플리케이션을 하드웨어에 독립적으로 만든다.
- 사이클 런타임 시스템(Cyclic runtime system) 동작을 보장한다.
- 다른 사이클 타임 설정을 허용한다.
- 8 가지 다른 테스크 클래스(테스크 claaes)를 제공한다.
- 시간 위반에 대한 반응을 보장한다.
- 모든 테스크 클래스에 대해서 여유시간(tolerance) 제한을 설정할 수 있다.
- IEC 61131-3 에 따라 광범위한 라이브러리를 제공한다.
- 모든 네트워크와 버스 시스템에 접속할 수 있다.
- 통합된 웹, FTP 그리고 VNC 서버를 포함한다.
- 종합 시스템 진단(SDM)을 제공한다.

Real-time operating system ₩ Method of operation

2.2 타겟 시스템

Automation Runtime 은 다양한 하드웨어 플랫폼에서 실행할 수 있다. X20 컨트롤러, 파워 패널(Power Panel), 그리고 Automation PC, PC 기반 하드웨어 플랫폼에서 사용된다.

그림 5: SG4 타겟 시스템

이 교육 자료는 B&R 타겟 시스템의 기능과 설정 옵션을 설명한다.
추가적인 타겟 시스템에 대한 정보는 도움말에서 찾을 수 있다.

Real-time operating system ₩ Target systems

2.3 설치와 시작

모든 타겟 시스템[1]은 플래시 메모리에서 Automation Runtime 과 Automation Studio 프로젝트를 부팅한다.

부팅가능한 CompactFlash 카드는 첫 번째 사용하는 Automation Studio 를 설정한다.

이 과정동안, CompactFlash 카드는 어플리케이션의 요구사항에 따라 나눠진다.

Automation Runtime 과 Automation Studio 프로젝트는 그 후에 CompactFlash 카드에 설치된다.

내부 어플리케이션 메모리가 있는 장치들은 이더넷 연결 또는 USB 원격 설치 메카니즘을 사용해서 업데이트한다.

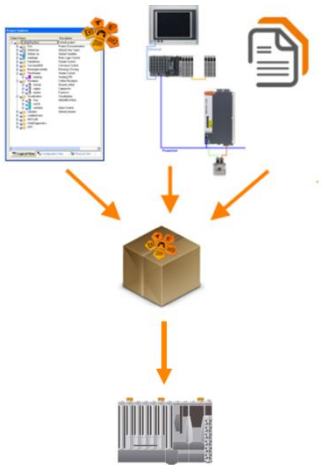


그림 6: 설치와 시작

[1]몇몇 타겟 시스템들은 장치 내부에 어플리케이션 메모리가 있다. 이 경우에 Automation Runtime, 어플리케이션 프로그램은 내부 어플리케이션 메모리로에서 불러온다. 시리얼 인터페이스, 이더넷 인터페이스 또는 USB 원격 설치 메카니즘을 사용해서 Automation Runtime 을 시스템에 설치한다.

예외적으로 PC 기반 타겟 시스템은 Automation Runtime for Windows (ArWin)이다. 이 경우, Automation Runtime 은 처음에 Automation Studio DVD 에 포함된 특별한 설치 툴을 사용해서 설치해야 한다.

2.3.1 가동시킬 CompactFlash 생성하기

Automation Studio 메인메뉴 <Tools> / <Create CompacFlash> 를 선택하면 ComopactFlash 카드 데이터가 생성된다.

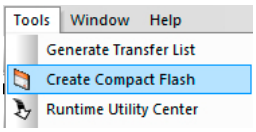


그림 7: Compact-Flash 데이터 생성하기

시작 후에, 프로젝트가 에러 없이 성공적으로 빌드되었는지 확인한다. 만약 그렇지 않다면, 리빌드(Rebuild)한다.

CompactFlash 카드를 설정하기 위해서는 CompactFlash 카드 리더기가 필요하다.

그 다음, CompactFlash 카드는 옆에 보이는 창에서 선택하면 된다. 사용자가 만족하는 셋팅이 만들어지면, CompactFlash 데이터 생성 과정이 시작된다.

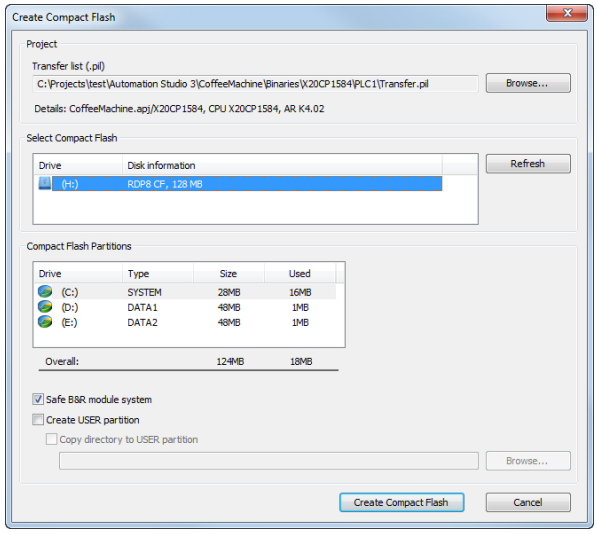


그림 8: CompactFlash 를 생성하기 위한 다이얼로그 박스

실시간 생산 시스템에는 Safe B&R module system 옵션 사용을 추천한다.

Diagnostics and service ₩ Service tools ₩ Runtime Utility center ₩ Creating a list ₩ data metium ₩ CompactFlash functions ₩ Creating a CompactFlash

2.3.2 온라인 상에서 설치

Automation Runtime 은 온라인 상에서 전달하는 것이 가능하다. 이를 위해서 온라인 인터페이스는 정확하게 설정되어야하고 올바른 모드에 있어야 한다.(4.4.1 “Startup”).

SNMP[2]를 포함한 타겟 시스템 브라우징은 SG4 시스템, 디폴트 런타임 V3.06[3] 이상에서 지원한다.

많은 타겟 시스템은 리셋버튼을 사용할 수 있다. 이 리셋 버튼을 사용하면, 그 시스템은 재시작되고 작동 모드가 변경될 수 있다. 각각의 사용자 메뉴얼에서 어떻게 컨트롤 시스템을 BOOT 모드로 변경하는지 찾을 수 있다.

연결망 구성

Automation Studio 에서 "Browse for target system"을 사용하면 컨트롤러에 연결할 수 있다. 이 기능은 B&R 컨트롤러에 대한 네트워크를 찾는다. 또한 검색 창에서 컨트롤러 연결에 대한 셋팅을 임시적으로 변경 할 수 있다.

Programming ₩ Building and transferring projects ₩ Establishing a connection oto the target system ₩ Ethernet connections ₩ Browse for targets

Automation Runtime 전달하기

Automation Runtime 은 연결망이 구성되면 전송 할 수 있다.

Programming ₩ Building and transferring projects ₩ Online services ₩ Transfer Automation Runtime ₩ Transferring to SGx target systems ₩ Installing via an online connection

2.3.3 USB 원격 설치를 이용한 설치 하기

Automation Runtime 과 어플리케이션 소프트웨어는 USB 를 사용해서 타겟 시스템에 다운로드 할 수 있다.

이를 위해, USB 원격 설치 기능이 사용할 수 있도록 컨트롤러 시스템 셋팅에서 설정해야한다.

먼저 Automation Studio 에서 프로젝트를 컴파일하여 전송할 목록을 만든다.

원격 설치구성은 'Runtime Utility Center'를 통해서 USB 로 복사된다. Runtime Utility Center 는 Automation Studio 가 설치될 때 자동으로 설치된다.

USB 를 타겟 시스템에 연결한다.

컨트롤러가 부팅시 필요하다면 Automation Runtime 버전과 어플리케이션 소프트웨어를 확인한 후 업데이트한다.

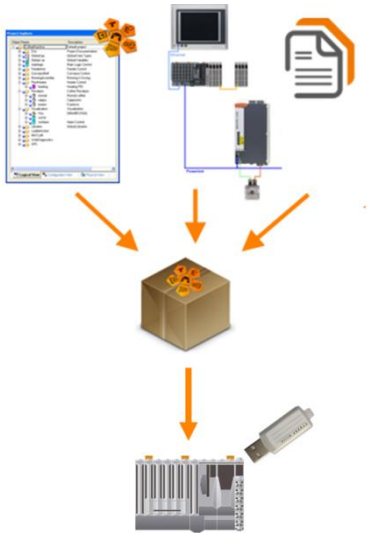


그림 9: Automation Runtime 과 어플리케이션은 USB 원격 설치를 이용해서 업데이트된다.

[2] 간단한 네트워크 관리 프로토콜은 네트워크 중앙에서 디바이스를 모니터링하고 제어하기 위해 사용되는 네트워크 프로토콜이다. (예를들어 라우터, 서버, 스위치, 프린터, 컴퓨터 등)

[3]컨트롤러에 미리 설치된 디폴트 런타임은 Automation Runtime 의 축소된 변형이다. 예를 들어서 CompactFlash 카드로부터 실제 부팅 과정을 다루는 책임이 있다.

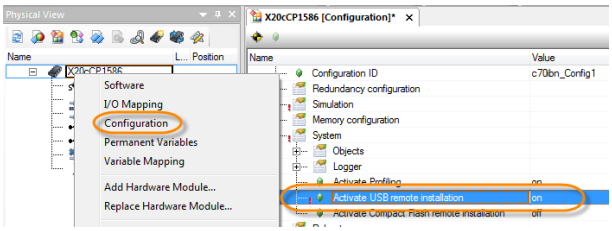


그림 10: System propertes 에서 USB 원격 설치 사용하기

USB 원격 설치 사용하기

시스템 설정에서 이 기능을 활성화 하거나 컨트롤러에서 기능을 요구한다면 컨트롤 시스템은 맨 처음 USB 원격 설치 이후에 이 기술을 지원하도록 설정해야 한다. 이 구성은 Physical View 에서 컨트롤러의 단축메뉴를 사용하여 열 수 있다. 요구된 설정 목록은 “System” 항목에 있다.

전달 목록 생성하기

먼저 Automation Studio 에서 전달 목록을 생성한다. 이 과정은 프로젝트를 컴파일할때 자동으로 실행된다. 또는 <Tools> / <Generate transfer list>를 선택해서 실행시킬 수 있다. 전달 목록은 컨트롤 시스템 설치를 위한 모든 데이터이다.

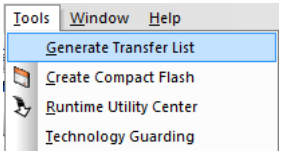


그림 11: 전달 목록 생성하기

USB 원격 설치 구성 생성하기

서비스 툴 Runtime Utility Center 는 Automation Studio 매인 메뉴 <Tools> / <Runtime Utility Center>를 선택해서 시작한다. “Create a remote install structure”는 선택 다이얼로그 박스에서 선택해야 한다.

USB 는 설정 다이얼로그 박스에서 선택할 수 있다. 또한 어플리케이션에 따라 Automation Runtime 이 업데이트가 필요하다면 이 또한 선택할 수 있다. 버전 확인은 타켓 시스템이 의도치 않게 업데이트 되는 것을 보호한다.

“Start” 버튼을 누르면 원격 설치 구성이 USB 에 복사된다. 시스템은 새롭게 설정된 USB 을 이용해서 업데이트 될 수 있다.

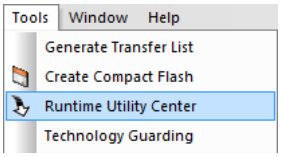


그림 12: Runtime Utility Center 시작

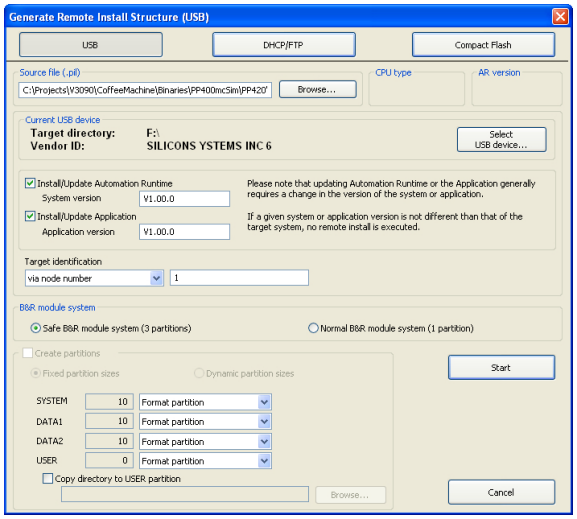


그림 13: Runtime Utility Center 설치 dialog

많은 타겟 시스템들은 리셋버튼을 사용할 수 있다. 이 리셋 버튼을 사용하면 시스템이 재시작되고 작동중인 모드를 변경 할 수 있다. 해당 사용자 메뉴얼에서 어떻게 컨트롤 시스템을 BOOT 모드로 변경하는지 확인 할 수 있다.
온라인 연결을 이용하여 BOOT 모드에서 Automation Runtime 을 설치할 수 있다.

Diagnostics and service ₩ sercive tool ₩ Runtime Utility Center ₩ Creating a list ₩ data medium ₩ Remote install structure creation

2.4 Automation Runtime 변경과 업데이트

Automation Runtime, 비주얼 컴포넌트, 모션, CNC 런타임 환경을 위한 새로운 소트트웨어 버전이 사용 가능하다면, Automation Studio 프로젝트에서 이 소프트웨어 버전을 변경하고 설정 할 수 있다.

런타임 환경을 위한 소프트웨어 버전은 메인메뉴 **<Project> / <Change runtime versions>**을 누르면 변경할 수 있다.

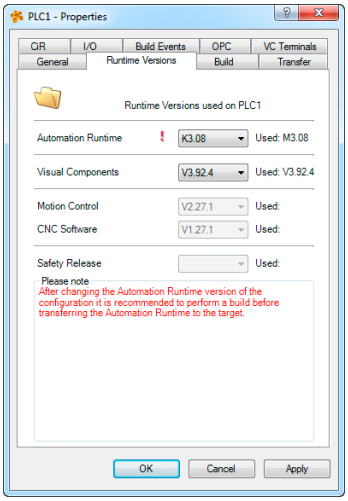


그림 14: 런타임 버전 변경

Projecti management ₩ Changing runtime versions

모션 제어, CNC 소프트웨어, 비주얼 컴포넌트 버전을 변경하는 것은 Logival View 구성에 사용되는 단일 인스턴스 라이브러리가 존재하기 때문에 모든 하드웨어 구성에 영향을 준다.

Automation Runtime 버전이 변경된 이후에는 프로젝트를 리빌드 후 전송해야 한다.

Automation Stuido 에서 업그레이드하기

Automation Runtime 이 타겟 시스템에 설치되어 있으면 그 프로젝트는 Automation Runtime 의 새로운 버전을 **온라인**으로 전송 할 수 있다.

이더넷 연결을 사용할 때, 새로운 Automation Runtime 버전이 전송될 때 UseROM 이 삭제되기 때문에 **sysconf** 모듈이 **SystemROM** 타겟 메모리(디폴트)에서 사용될 수 있도록 하라. UserROM 이 삭제된 후에는 더 이상 제어기와 연결 할 수 없다. 소프트웨어 모듈을 위한 타겟 메모리는 소프트웨어 구성(software configuration)에서 설정 할 수 있다.

타겟 시스템에 프로젝트를 전송할때, 버전 충돌이 발생되면 다이얼로그 박스에 표시된다.

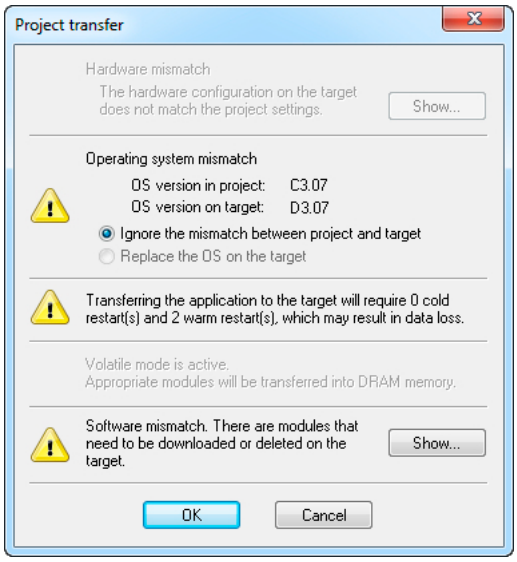


그림 15: 버전 충돌 발견

Automation Studio 없이 업그레이드하기

Automation Studio 를 사용한 업그레이드가 불가능하다면 Runtime Utility Center 를 사용해서 완전한 설치 이미지를 생성할 수 있다. CD 또는 USB 플래시 드라이버를 사용하여 전송할 수 있다.

업데이트를 위해서 타겟 시스템에 온라인 연결이 된 PC 가 필요하다.

Diagnostics and service ₩ Service tool ₩ Runtime Utility center ₩ Creating a list ₩ data medium ₩ Generating an installation package

3 메모리 관리

Automation Runtime 타겟 시스템의 메모리는 RAM 과 ROM 으로 구성된다.

각 파트는 Autmation Runtime 런타임 전용으로 사용된다; 나머지는 어플리케이션에 사용 될 수 있다.

3.1 플래시 메모리의 지역 그룹화

빌드되는 동안, Automation Studio 프로젝트는 Automation Runtime 에 의해 실행될 수 있는 확장자가 ".br" 인 모듈을 생성한다.

소프트웨어 구성(Software configuration)에서, 각 모듈은 자동으로 타겟 메모리(User ROM, System ROM 으로 나눠지는)에 할당된다. 데이터가 같은 데이터 저장 매체에 저장되기 때문에 순전히 지역 그룹화이다.

User ROM 은 Automation Studio 프로젝트를 위한 모든 BR 모듈이 저장되는 CompactFlash 카드 메모리이다.

System ROM 은 Automation Runtime 과 프로젝트를 위한 시스템 모듈이 저장되는 CompactFlash 카드[4] 메모리이다.

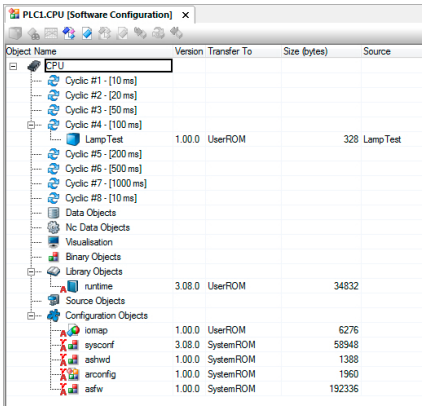


그림 16: .br 모듈을 위한 타겟 시스템

그림 17: CompactFlash = System ROM + User ROM

[4] User ROM 과 System ROM 의 논리 구조는 통합 어플리케이션 메모리가 있는 시스템에 적용됩니다.

3.2 온라인 소프트웨어 비교

활성화된 온라인 연결이 있을 때, Automation Studio 에 설정된 모듈과 타겟 시스템에 설치된 모듈을 비교할 수 있다.

소프트웨어 비교는 메인메뉴 <Online> / <Compare> / <Software> 를 선택하면 실행할 수 있다.

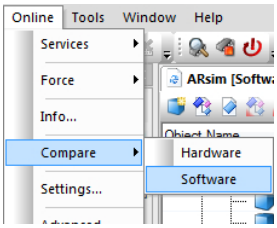


그림 18: 소프트웨어 비교 열기

소프트웨어 비교 창은 프로젝트뿐 아니라 컨트롤러에 있는 모듈 전체를 보여준다. “Size (bytes)” 열은 타겟 시스템에 있는 BR 모듈의 타겟 메모리를 보여준다. 버전, 사이즈, 타겟 메모리 그리고 모듈의 빌드 날짜도 비교가능하다.

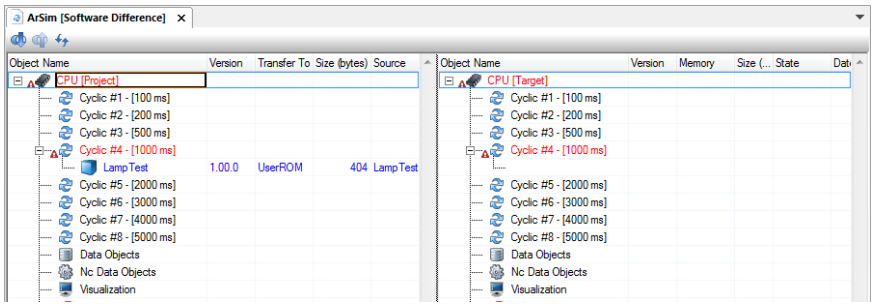


그림 19: 온라인 소프트웨어 비교 창

예제: User ROM 에 있는 모듈 삭제하기

메인메뉴의 <Online> / <Services> / <Clear Memory> 를 선택해서 User ROM 에 데이터를 삭제한다.

그 후 온라인 소프트웨어 비교를 사용해서 컨트롤러에 어떤 대상이 남아있는지 확인하라.

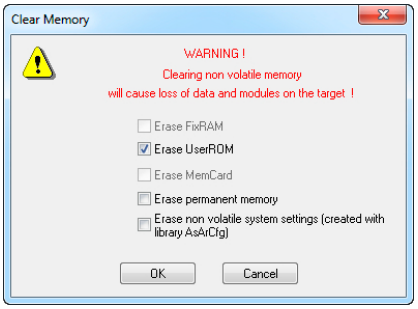


그림 20: User ROM 에서 선택된 메모리를 삭제하기 위한 창

System ROM 만 타켓 메모리로 정의된 모듈에 표시됩니다.

이더넷 구성 같은 시스템 설정이 타켓 메모리로 설정된 System ROM 에 “sysconf.br”모듈이 저장되고, 온라인 연결은 User ROM 이 삭제될 때 손실되지 않는다.

User ROM 이 삭제될 때, 시스템은 자동으로 재시작하고 진단모드(diagnostic mode)로 전환된다. 이는 온라인 연결을 막는다.

3.3 사용되는 RAM

RAM 은 Automation Runtime 을 실행하기 위한 데이터가 있고 Automation Studio 어플리케이션을 불러오고 실행할 수 있도록 빠르게 데이터를 읽고 쓸 수 있는 메모리이다.

타겟 시스템은 항상 DRAM 을 가지며, SRAM 또는 FRAM 은 선택사항이다.

그림 21: DRAM, SRAM 그리고 FRAM

DRAM: DRAM 은 시작 후에 정의되지 않은 상태에 놓여있는 RAM 메모리이다.
부팅 후 Automation Runtime 은 모든 필요한 BR 모듈을 DRAM 에 불러오고, 빠르게 접속할 수 있도록 한다. RAM 안에 있는 BR 모듈은 Automation Studio 안에 설정된 지역 또는 전역 변수 메모리 저장 장소가 필요하고(3.5.3 변수 메모리 초기화) 이 메모리 저장장소를 책임지고 초기화해야한다.

SRAM: DRAM, SRAM (static RAM)과 달리 배터리에 의해 버퍼링된다. 따라서 배터리가 작동중이라면 파워가 꺼진 후에도 데이터가 유지될 수 있다.

FRAM: 새로운 컨트롤러 모델은 배터리 없이 생산된다. SRAM, DRAM 과 달리 FRAM 은 강유전성 특성을 지닌 크리스탈을 사용하여 비휘발성(전원이 끊겨도 데이터가 없어지지 않는) 전자 메모리 타입으로 메모리를 저장한다. 따라서 파워가 꺼져도 데이터를 유지할 수 있다. 이것은 배터리가 없어도 된다.

3.4 메모리 요구사항 결정을 위한 툴

다음의 방법을 이용하여 남아있는 메모리의 양을 확인할 수 있다:

- CPU 정보 읽어오기
- 시스템 진단 매니저(System Deiagnostics Manager)
- BRSystem 라이브러리에 있는 MEMxInfo() 펑션 블록

온라인 정보

메인메뉴 <Online> / <Info> 를 선택하면 사용가능한 메모리 양에 대한 일반적인 정보를 볼 수 있다. 이 옵션은 모든 타겟 시스템에 사용할 수 없다. 온라인 정보는 백업 배터리 상태와 컨트롤러의 현재 시간을 볼 수 있다.

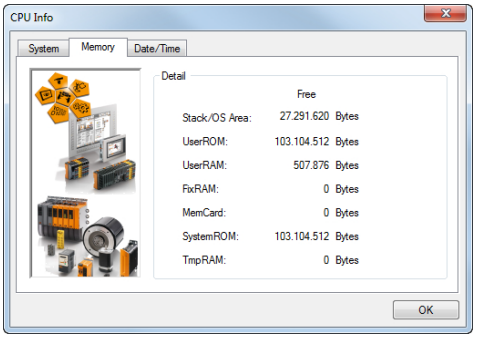


그림 22: CPU 정보 읽어오기

시스템 진단 매니저(System Diagnostics Manager: SDM)

시스템 진단 매니저(SDM)는 Automation Runtime V3.0 과 그 이상 버전에 있는 통합 컴포넌트이다.
메인메뉴 <Tools> / <System diagnostics Manager>를 선택하면 브라우저 창에 SDM 을 열 수 있다.

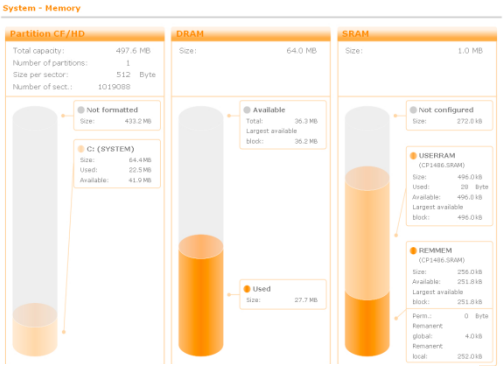


그림 23: SDM 에 있는 메모리 할당량

예제: 타겟 시스템에 이용 가능한 메모리 체크

TM210 에서 생성했던 프로젝트를 타겟 시스템에 전달한다.

이더넷 연결을 사용해서 타겟 시스템과 Automation Studio 을 연결한다. 앞에서 설명했던 방법 중 하나로 타겟시스템의 메모리 할당량을 확인하라.

사용중인 타겟 시스템에 존재하는 메모리 타입을 알 수 있고 어떻게 사용되는지 확인 할 수 있다.

Diagnostics and service ₩ Diagnostic tools ₩ Information about the target system

Diagnostics and service ₩ diagnostics tools ₩ System diagnostics Manager

Programming ₩ Libraries ₩ Configuration, system information, runtime control ₩ AsHW

3.5 전역 그리고 지역 변수

변수들은 데이터 타입에 의해서 사이즈와 구성이 결정되는 심볼릭 프로그래밍 요소(Symbolic programming elements)이다. 프로젝트가 빌드될 때, 변수들은 메모리의 특정 장소에 할당된다.

변수의 범위와 특성은 시작과 런타임 동작시 결정된다.

Programming ₩ Variables and data types

3.5.1 지역 변수

지역 변수는 특정한 프로그램 안에서만 사용할 수 있고 다른 프로그램에서는 사용할 수 없다.

지역 변수는 프로그램과 같은 레벨에 있는 “.var” 파일에서 관리된다.

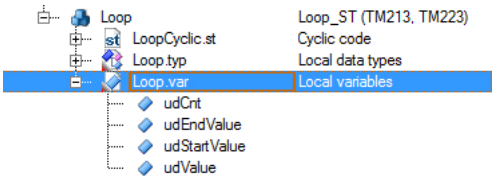


그림 24: “Loop“ 프로그램 에 있는 지역 변수

예제: 지역 변수를 사용하여 “Loop” 프로그램 생성하기

프로젝트 Logical View 아래에 보이는 지역변수을 포함한 “Loop” 라는 Structured Text 프로그램을 생성하라.

UDINT 타입의 “udCnt”, “udValue”, “udStartValue”, “udEndValue” 라 명명한 변수를 추가하라.

프로그램 사이클 부분(cyclic)에 루프를 생성해라. 이 루프는 뒤에 나오는 예제에서 설명되고 사용될 것이다.

```
PROGRAM _CYCLIC
        FOR udCnt := unStartValue TO udEndValue DO
                udValue := unValue + 1;
        END_FOR
END_PROGRAM
```

1) 프로그래밍 언어로 “Structured Text”을 선택
2) “Loop” 프로그램 추가
3) “Loop.var”파일을 열고 변수 추가
4) “Loop.var”파일을 저장한 후, “LoopCyclic.st”에 있는 변수들은 테스크의 프로그램에서 사용될 수 있음

프로그램을 추가시, 소프트웨어 구성(software configuration)에 자동으로 할당된다.

예제: 소프트웨어 구성(software configuration)에서 프로그램 다중 할당

마지막 작업을 완료한 후, 드래그-앤-드롭(drag-and-drop)으로 Logical View 에서 소프트웨어 구성(software configuration)으로 동일한 프로그램을 두 번 이동시켜라.

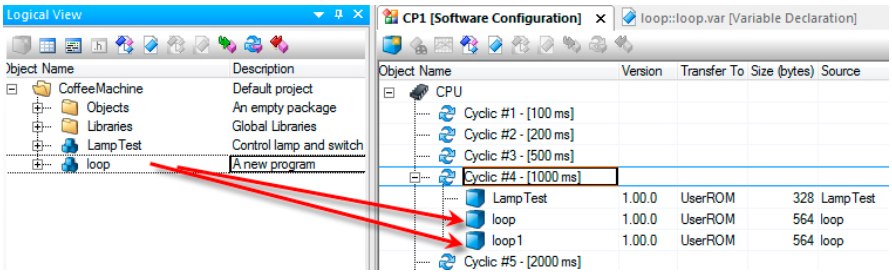


그림 25: 프로그램의 다중 할당

1) 소프트웨어 구성(software configuration) 열기
2) Project Explorer 안에 있는 Logical View 을 선택
3) "Loop" 프로그램을 드래그-앤-드롭으로 소프트웨어 구성으로 이동 시키기

"Loop" 프로그램의 두 인스턴스를 위한 Watch 창은 Logical View 또는 소프트웨어 구성에서 단축 메뉴를 이용해서 열 수 있다. 선택 다이얼로그 박스는 당신이 모니터링 하고 싶은 변수와 관련된 내용을 Logical View 의 Watch 창에서 볼 수 있다.

Watch 창은 <CTRL + W>를 눌러서 열 수 있다.

"Loop"와 "Loop1"의 모든 변수들이 Watch 창에 추가되면, 나눠진 변수는 해당되는 테스크에서만 볼 수 있다.

한 테스크에서 지역 변수를 변경하면 오직 그 테스크 안에서만 영향을 끼친다 – 다른 테스크의 지역 변수 값은 변경되지 않는다.

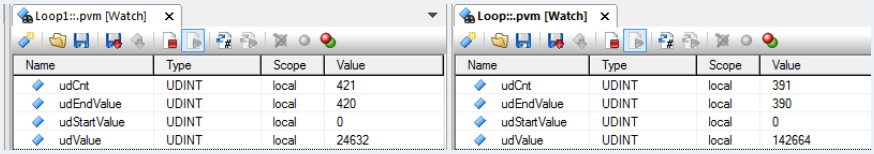


그림 26: 두 테스크(task)의 Watch 창

변수 파일에는 포함되어 있지만 프로그램에서 사용되지 않은 변수들은 타겟 시스템에 사용할 수 없다.

빌드가 진행되면서 경고 창이 나타난다.

Warning 424: Variable <variable name> is daclared but not being used in the current configuration.

Programming ₩ Variables and data types ₩ Scope of daclarations

3.5.2 전역/ 패키지-전역 변수

전역 변수는 Logical View 의 가장 윗부분에 보인다. 전역 변수들은 Automation Studio 프로젝트 전체에 사용할 수 있다. 그 이유는 이 변수들은 모든 프로그램에서 사용될 수 있기 때문이다.

전역 변수는 최상단에 있는 Global.var 파일에서 관리된다. 또한 Additional.var 파일은 그 프로젝트를 더 편리한 구조로 바꾸기 위해 만들 수 있다.

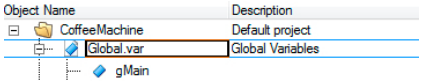


그림 27: 전역 변수

패키지-전역 변수는 패키지 안에서 명시되고 각각의 패키지, 종속 패키지 안에서만 사용할 수 있다.

전역 변수는 몇몇 프로그램 사이에 데이터 교환이 필요할때 사용된다. 다른 대안은 프로그램의 지역 변수를 변수 매핑을 이용해서 연결하는 것이다.

예제: "gMain" 전역 변수를 생성하고 "Loop" 프로그램에서 사용하기

"Global.var" 에 새로운 전역 변수를 선언하고 UDINT 데이터 타입과 "gMain" 라는 이름으로 할당하라.

이 변수는 "Loop" 프로그램에서 반복적으로 증가되어야 한다.

```
gMain := gMain + 1;
```

1) "Global.var" 파일 열기
2) UDINT 타입의 변수 "gMain" 추가
3) "Global.var" 파일을 저장하면, "LoopCyclic.st"안에 있는 변수는 프로그램 안에서 사용될 수 있음

만약 "gMain" 변수가 "Loop" 그리고 "Loop1" 테스크 Watch 창에 추가된다면, 둘 중 하나의 테스크에서 값을 입력하면 다른 창에서도 값이 변할 것이다.

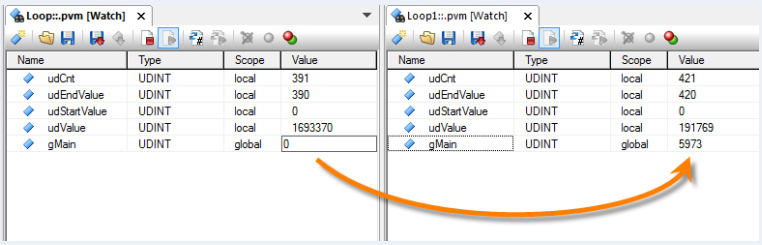


그림 28: 전역 변수에 값 입력하기

Programming ₩ Variables and data types ₩ Scope of daclarations
Programming ₩ Editors ₩ Configuration editors ₩ Variable mapping

3.5.3 변수 메모리 초기화

디폴트로, 초기화 과정 동안 변수에 "0"이 입력된다. 그러나 다른 초기화 값은 변수선언에서 명시된다.

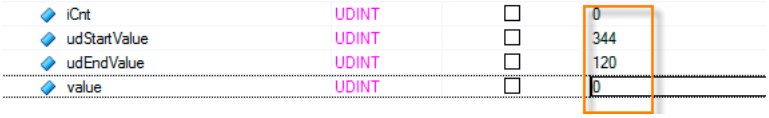


그림 29: "Loop.var" 에서 변수 초기화

초기화는 아래에 나오는 **"LoopInit.st"** 초기화 서브루틴에 아래의 코드와 같이 나타낼 수 있다:

```
PROGRAM _INIT
        udEndValue := 50;
END_PROGRAM
```

예제: "udEndValue" 변수 초기화
"Loop" 프로그램 안에 **"udEndValue"** 변수를 50 으로 초기화해라. Watch 창에서 그 변수를 확인하라.

1) **"Loop.var"** 파일 열기.
2) 변수 **"udEndValue" 의 "Value"** 열 안에 초기값 설정하기.

"Loop"와 "Loop1" 테스크의 Watch 창에서 변수 "udEndValue"를 50 으로 초기화하라.
런타임 동안, 이 값은 어떤 다른 값으로 변경 할 수 있다.

3.5.4 상수(Constant)

상수(Constant)는 프로그램이 작동하는 동안 절대 변하지 않는 값을 가지는 변수이다.

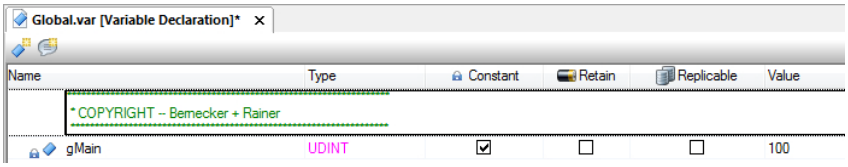


그림 30: 상수 선언

예제: 상수로 선언된 전역 변수 "gMain"

100 을 가지는 상수로 전역 변수 **"gMain"** 을 설정하라.

1) "Global.var" 파일 열기
2) 변수 **"gMain" 의 "Value"** 행에 값 설정하기
3) 변수 **"gMain"** 를 상수로 선언 하기

"gMain" 이 사용되고 나서, 빌드 과정중에 에러 메세지가 나타날 수 있다.

이것은 상수가 허용되지 않아서이다! 에러 없이 프로그램을 작동시키기 위해서는 변수 "gMain"이 상수로 선언되는 것을 허용하지 않는다.

LoopCyclic.st (Ln: 19, Col: 8) : Error 1138: Cannot write to variable 'gMain'.

Programming ₩ Variables and data types ₩ Variables ₩ Constats

3.5.5 Retain 변수의 시스템 동작

타겟 시스템이 부팅될 때, Automation Runtime 은 자동으로 시작되고 모든 부팅 단계(boot phases)를 걸쳐서 작동한다 (4.1 "Automation Runtime 시작하기").

시스템 재시작(warm restart) 이후에도 프로세스 변수(Process variables)가 유지되기 위해서, 변수들을 battery-backed remanent memory 에 저장해야하고 시작할 때 자동으로 불러오도록 해야한다.

비휘발성(remanent) 메모리에 변수를 저장하기 위해서 다음 요구사항을 타겟 시스템과 변수 선언에서 만족시켜야 한다.

- Battery-backed SRAM 또는 FRAM 이 있는 타겟 시스템
- "Retain" 으로 설정된 변수

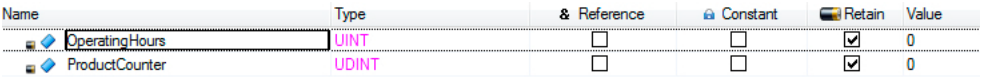


그림 31: Battery-backed 변수들

다른 타겟 시스템들은 SRAM 또는 FRAM 이 사용가능한 다양한 제품군이 있다.

SRAM 또는 FRAM 의 이용 가능성이나 사이즈에 대한 정보는 각각의 하드웨어 문서에서 찾을 수 있다.

Retain 변수 사용 예시:

- 작동중인 타임 카운터
- 에러난 제품 개수
- 제품 ID
- 카운터 타입
- 전원이 연결되지 않았을 때 기계의 상태와 상황
- …

프로그램이 시작되거나 (e.g. 설정 데이터) 또는 변경사항이 있을 때(e.g. 레시피) 오직 읽거나 쓸 수만 있는 데이터는 플래시 메모리에 저장되어야 한다.

Warm restart

재시작(Warm restart) 은 다음 작업에 의해 작동합니다:

- 전원을 끈 후에 다시 전원을 킴.
- 시스템 설정을 변경하고 해당 데이터를 타겟 시스템에 전송.
- Automation Studio 에서 warm restart 을 명령 (전원 공급과 같은).

Retain 변수들은 **power-on** 이나 Automation Studio 에서 **warm restart** 되어도 값을 유지한다.
만약 시스템이 **cold Restart** 된다면, Retain 변수들은 "0"으로 초기화 된다.

Cold restart

강제 재시작(Cold restart)은 다음 작업에 의해 작동합니다:

- CompactFlash 카드 교환 후 재시작.
- UserROM 삭제 후 재시작.
- Automation Stidio 에서 Cold restart 을 명령.
- SRAM 안에 있는 Retain 변수 battery backing 에 결함이 있을 때 재시작.

Permanent variables(영구 유지 변수)

변수를 영구적으로 저장하기 위해서, permanent variables editor 에 추가하고 관리한다.

영구적으로 데이터를 저장하는 예시:

- 동작중인 타임 카운터
- Retain 변수가 지워질때, 플래시 메모리에 저장되지 않았지만 유지되야하는 모든 데이터

오직 **전역** Retain 변수만 Permanent variables(영구 유지 변수)로 설정할 수 있다.

영구 메모리는 시스템에 의해 절대로 포맷되거나 쓰여지지 않는다. 어플리케이션이 시작되는 동안 변수 값은 오직 사용자만이 변경할 수 있다.

적절한 기초 없이 영구 유지 변수는 프로그램에 부합하지 않거나 변경될 수 있다.

이는 CPU 또는 배터리가 변경시 반드시 기억해야 할 사항이다.

어떤 컨트롤러에 항상 저장되어야 하는 영구 변수들을 사용하는것 대신에 Technology Guard 에 데이터를 저장할 수 있다. AsGuard 라이브러리를 사용하면 Technology Guard 의 데이터 저장 장소에 접근할 수 있다.

Programming ₩ Variables and data types ₩ Variables ₩ Variable remanence

Programming ₩ Editors ₩ Configuration editors ₩ Permanent variables

Automation software ₩ Technology Guarding

Programming ₩ Libraries ₩ Configuration, system information, runtime control ₩ As-Guard

4 Runtime performance

이번 장에서는 Automation Studio 어플리케이션의 런타임 작동과 Automation Runtime 에 대해 설명할 것이다.

Automation Studio 안에서 사용할 수 있는 몇 가지 진단 툴을 통해 어떻게 작업이 이뤄지는지 알 수 있을 것이다.

4.1 Automation Runtime 시작하기

Automation Runtime 은 타겟 시스템에 전원이 공급되면 부팅한다.

이 과정 동안 아래의 작업이 실행된다:

- 하드웨어 확인
- 필요한 하드웨어/펌웨어 업그레이드
- BR 모듈 확인
- ROM 에서 DRAM 으로 BR 모듈 복사
- Retain 변수를 DRAM 으로 복사
- 변수 메모리 초기값 설정
- 서브루틴 초기화 실행
- 사이클 프로그램 활성화

Real-time operating system ₩ Method of operation ₩ Boot behavior

Real-time operating system ₩ Method operation ₩ Module ₩ Data security

부팅 단계에서 에러가 발생하지 않았다면 타겟 시스템은 RUN 모드로 시작한다.

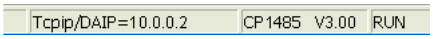


그림 32: RUN 모드 상태의 X20CP1485

에러 상태 정보를 얻으려면, “TM233-Automation Studio Diagnostics” 교육 자료를 참고하라.

Automation Runtime 작동 상태

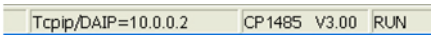
부팅 과정은 네 단계로 나눠진다.

첫 번째 단계가 완료되면, 다음 단계를 수행한다.

“RUN” 단계에서, Automation Runtime 은 Automation Studio 에서 생성된 어플리케이션을 시작하고 이를 반복 실행한다.

그림 33: 부팅 단계

일부 이벤트는 타켓 시스템이 단계 중 하나를 종료하고 진단 목적을 위해 각각의 시스템 상태에서 대기하도록 야기한다. 다음 표에 자세한 설명이 있다.:

작동 상태	이 시스템 상태가 발생하는 상황
BOOT(부팅)	• CompactFlash 가 삽입되지 않았을 때 • CF 카드에 운영 체제(operating system)이 없을 때 • 노드 번호가 “00”[5]으로 설정됐을 때
DIAGNOSTICS(진단)	• 메모리 초기화 • 심각한 시스템 에러 • 노드 번호가 “FF”로 설정됐을 때
SERVICE(서비스)	• 0 으로 나눴을 때 • 페이지 결함 • 사이클 타임 위반 • Automation Studio 에 의해 CPU 가 중단되었을 때 • 그 외의 에러
RUN(동작)	• 에러가 없을 때  그림 34: RUN 모드

Real-time operating system ₩ Method of operation ₩ Operating status

Automation Runtime 부팅 단계

컨트롤러가 부팅 되는 중에, RUN 상태에 도달하기 전에 중간 단계를 통해 동작할 수 있다. 또한 이 상태는 Automation Studio 상태 바(status bar)에 나타난다.이 단계들은 일반적으로 매우 짧다.

시스템 상태	설명
STARTUP	이 단계에서는 운영 체제와 관련된 과정이 초기 절차가 수행된다.
FIRMWARE	이 단계에서는 펌웨어를 업데이트한다.
INIT	이 단계에서는 어플리케이션의 초기화 서브루틴이 작동한다.

X20 시스템일때, RUN 상태전의 단계(STARTUP, FIRMWARE, INIT)에는 R/E LED 가 초록색으로 깜빡거린다.

Real-time operating system ₩ Method of operation ₩ Boot phases

[5] 사용되고 있는 장비에 따라서 노트 선택 스위치는 작동 모드로 설정되거나 전용 작동 모드 스위치가 있다. 추가적인 정보는 사용자 매뉴얼에서 확인 할 수 있다.

4.2 프로그램 초기화

프로그램은 초기화 서브루틴(처음에 스캔되는)이 있다.

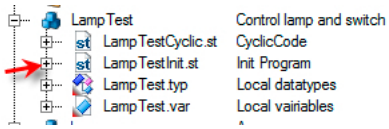
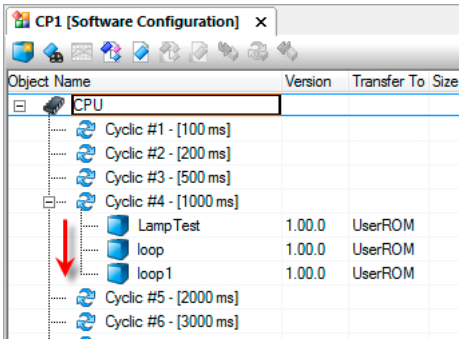


그림 35: 프로그램 초기화

초기화 서브루틴은 변수를 초기화 시키고 PLC 데이터와 사이클 서브루틴에서 실행될 기계 설정을 읽어온다.

4.2.1 초기화 서브루틴의 동작

첫 번째 사이클 프로그램이 실행되기 **이전에**, 모든 초기화 서브루틴은 소프트웨어 구성에 명시되기 위해 **한 번** 실행된다.



왼쪽 그림에서, 초기화 서브루틴은 아래의 순서에 따라 실행된다:

1 "LampTest" 테스크를 위한 초기화 서브루틴
2 "Loop" 테스크을 위한 초기화 서브루틴
3 "Loop1" 테스크을 위한 초기화 서브루틴

그림 36: 초기화 서브루틴의 실행

초기화 서브루틴은 사이클 타임 모니터링의 대상이 아니기 때문에, 초기화과정이 오래걸리더라도 오류가 나지 않을 것이다.

Real-time operating system ₩ Target systems ₩ SG4 ₩ Runtime performance ₩ Starting initialization subroutines

4.2.2 초기화 서브루틴에서 함수 호출

함수를 사용할 때, 함수가 호출될 때 리턴하는 값은 중요하다.

여러번 호출되야하는 함수들은 프로그램 사이클에서 호출할 수 있다.

4.3 사이클 프로그램 시퀀스

프로그램 또는 테스크 클래스는 소프트웨어 구성에서 특정한 실행 시간에 할당된다.

소프트웨어 구성에 있는 프로그램들은 테스크라 부른다. 컨트롤러는 소프트웨어 구성에 할당된 프로그램만을 실행시킨다.

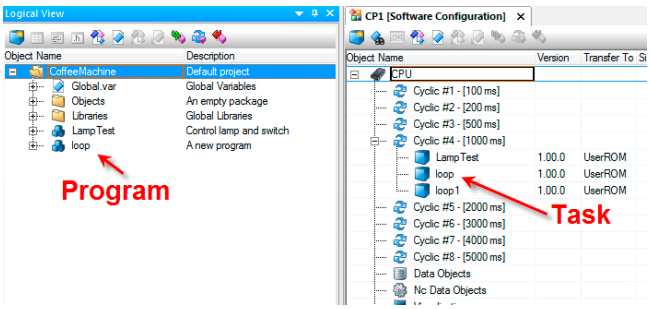


그림 37: 프로그램과 태스크(Task)의 관계

4.3.1 운영 체제 멀티 태스킹 하기

Automation Runtime 은 **결정론적, 실시간 멀티태스킹** 운영 체제이다.
테스크는 서로 다른 **"time slices"** 후에 정의된 리소스에 할당되고 실행된다. 테스크 실행시간이 모두 동일하지 않지만, 테스크가 시작하는 순간뿐만 아니라 I/O 시스템이 접속될 때의 순간은 제 시간에 정의된다.

4.3.2 테스크 클래스와 사이클 타임(cycle time)

테스크는 테스크 클래스에 의해 정의된 시간에 반복적으로 실행된다 – 사이클 타임.

설정 가능한 사이클 타임은 최대 8 개까지 설정할수 있으며, 특정 요구 사항에 최적화 할 수 있도록 도움을 제공합니다. 테스크 클래스는 같은 사이클 타임, 우선 순위, 허용 오차(tolerance)을 가지는 테스크를 포함한다.

모든 테스크들이 같은 타임 프레임 안에서 동작할 필요는 없다. 빠르게 실행되야하는 제어 테스크는 낮은 번호 테스크 클래스에 할당해야 한다. 느린 과정들은 높은 번호 테스크 클래스에 할당해야 한다.

이번 예제는 100 milliseconds 의 사이클 타임을 가지는 테스크 클래스 #4(Cyclic #4)에 세 개의 테스크가 있다. 이 테스크 각각이 실행되는 시간을 무시한다면, 프로그램 시퀀스는 아래와 같이 수행된다.:

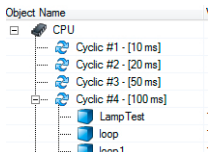


그림 38: 사이클 타임

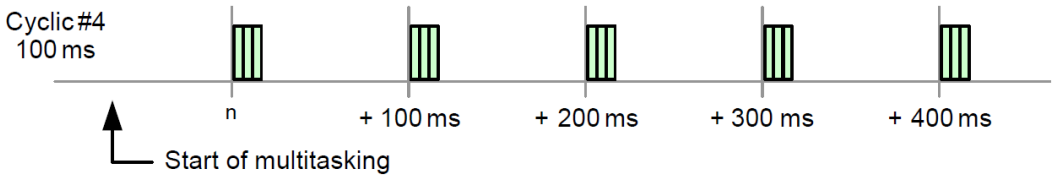


그림 39: 각각의 사이클 동안 다시 호출되는 테스크들

위의 그림은 테스크 클래스에 포함된 테스크들이 100 milliseconds 마다 실행되는 것을 나타낸다.

Real-time operating system ₩ Target systems ₩ SG4 ₩ Runtime performance ₩ Start of cyclic tasks

4.3.3 사이클 타임과 우선순위

테스크 클래스의 우선순위는 숫자로 결정된다.
낮은 숫자일수록 우선순위가 높은 테스크 클래스이다.
테스크 클래스 사이에서 테스크를 이동시켜서 우선순위와 사이클 타임을 변경한다.

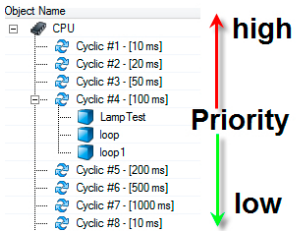


그림 40: 테스크 클래스 우선순위

테스크를 이동시켜도 테스크 실행시간은 변하지는 않는다. 주어진 시간 안에 테스크가 호출되는 순서만 변경된다.

예를 들어, "Loop" 테스크가 테스크 클래스 #4 에서 테스크 클래스 #1 으로 옮겨지면, 매 10 milliseconds 마다 실행된다.

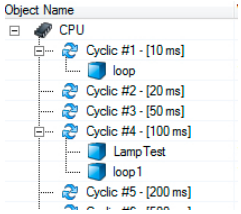


그림 41: 다른 테스크 클래스

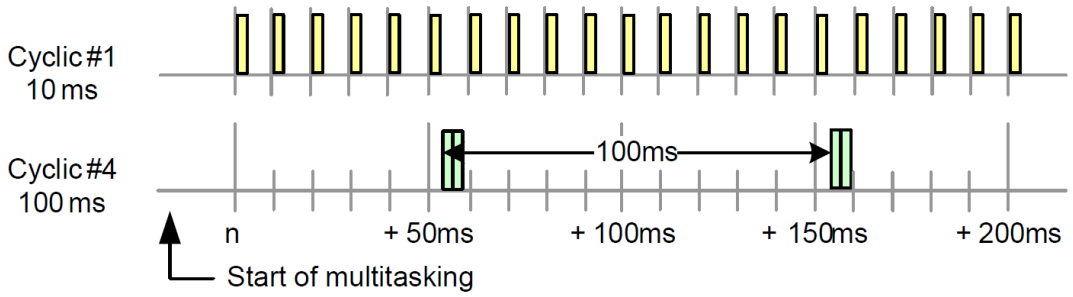


그림 42: 서로 다른 테스크 클래스에서 실행중인 테스크

테스크 클래스 #4 에 포함된 "LampTest"와 "Loop1" 테스크는 여전히 100 milliseconds 로 실행된다.

예제: 테스크클래스 #1 으로 "Loop" 테스크 이동하기

소프트웨어 구성에 있는 "Loop" 테스크를 **테스크 클래스 #1** 으로 옮겨야 한다.
"Loop"와 "Loop1"의 사이클 타임 변화가 같은 프로그램에 어떤 영향을 주는지 확인하다.

1) 소프트웨어 구성(software configuration) 열기
2) "Loop" 테스크를 테스크 클래스 #4 에서 테스크 클래스 #1 로 이동
3) 두 테스크의 Watch 창 열기
4) "udValue" 변수 변화 모니터링 수행

"Loop" 프로그램은 "Loop1"보다 10 배 더 빨리 실행된다. 그 결과 "udValue"가 더 많이 증가되었다.

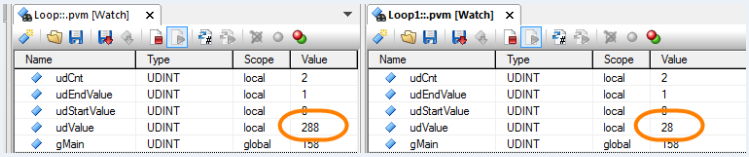


그림 43: 서로다른 테스크 클래스에서 사이클 타임

4.3.4 테스크 클래스 시작하기

멀티 태스킹 시스템이 시작된 후에 모든 테스크 클래스가 동시에 시작되는 것은 아니다.
시작 시점은 항상 테스크 클래스 사이클 / 2 이다.

예를 들어서, 100 ms 테스크 클래스는 50 ms 후에 시작될 것이다. 모든 테스크 클래스의 시작 시점을 나누는 것은 실행 횟수를 줄이고 프로세서에 높은 로드가 걸리더라도 출력에 흐트러짐을 최소한으로 만들기 위한 것이다.

Real-time operating system ₩ Target systems ₩ SG4 ₩ Runtime performance ₩ Start of cyclic tasks

4.3.5 Idle time

작동 중에, Automation Runtime 은 위에서 설명했던 테스크뿐만 아니라 서로 다른 **사이클(cyclic)** 시스템 테스크들을 동작시킨다. 이는 사용자에게 편리함과 안정성을 제공한다 (예를 들어서 모듈 확인, 온라인 커뮤니케이션).

사용하는 CPU 성능에 따라 테스크 실행 속도가 달라진다.

Automation Runtime 이나 사용자 테스크가 실행되고 있지 않는 시간을 **Idle time** 라고 한다.
Automation Runtime 은 아래의 업무를 Idle time 에서 수행한다.:

- 온라인 커뮤니케이션
- 화면 컴포넌트 어플리케이션 (Visual Components application)
- 파일 접근

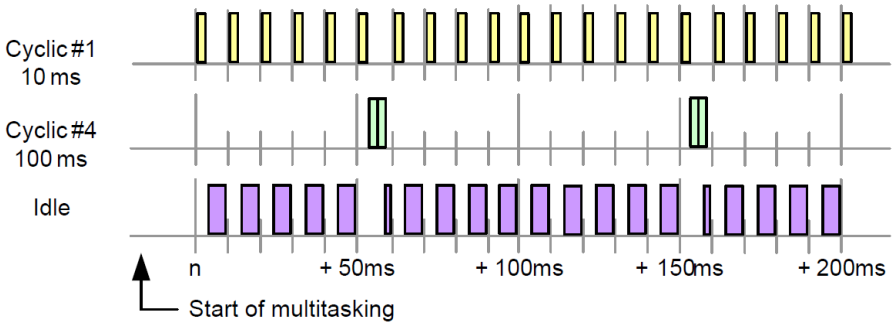


그림 44: 사이클 타임에서 Idle time

Idle time 이 화면 컴포넌트 어플리케이션을 실행시키기에 충분하지 않거나 안정적인 온라인 연결을 제공할 수 없다면, 테스크 클래스에 프로그램을 할당하거나 테스크 클래스의 사이클 타임을 조절해서 idle time 을 늘릴 수 있다.

프로파일러로 idle time 을 결정할 수 있다. 또한 System Diagnostics Manager 로 평균 시스템 로드를 확인 할 수 있다.

Real-time operating system ₩ Target systems ₩ SG4 ₩ Runtime performance ₩ Scheduling ₩ Idle time

4.3.6 I/O 맵핑 다루기 – I/O 스케줄러

I/O 스케줄러는 정확한 시간에 테스크 클래스를 시작하고 테스크 클래스에 있는 모든 테스크 실행에 대한 일관된 I/O 이미지를 제공한다.

이는 테스크 클래스가 시작할 때 입력과 테스크 클래스가 끝날 때 출력이 맵핑된 프로세스 변수에 즉시 전달된다.

각각의 테스크 클래스는 나뉘어진 I/O 이미지를 사용한다. 입력 상태는 테스크의 런타임 동안 변하지 않으며, 출력 상태는 테스크 클래스에서 그 테스크가 끝날 때까지 쓰여지지 않는다.

이미지에서 I/O 데이터 사이클

I/O 채널들은 사용하고 있는 I/O 필드버스 시스템에 설정 가능한 사이클을 이용해서 맵핑한다.

이 사이클은 각각의 필드버스 특성에서 설정한다. X2X 링크 인터페이스, 단축 메뉴에서 <Configuration>을 선택하라.

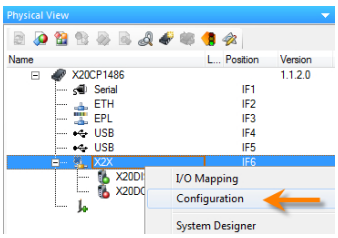


그림 45: X2X 링크 인터페이스의 특성 창

특성에서 설정한 사이클 타임은 I/O 데이터를 I/O 이미지에 복사하기 위한 기본 작업에 사용된다.

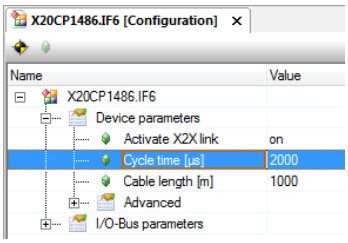


그림 46: X2X 사이클 타임 셋팅

필드버스 장치(이 경우에는 X2X 링크)는 설정된 사이클 타임 동안 I/O 이미지에 입력을 복사하고 I/O 이미지로부터 출력을 복사해온다.

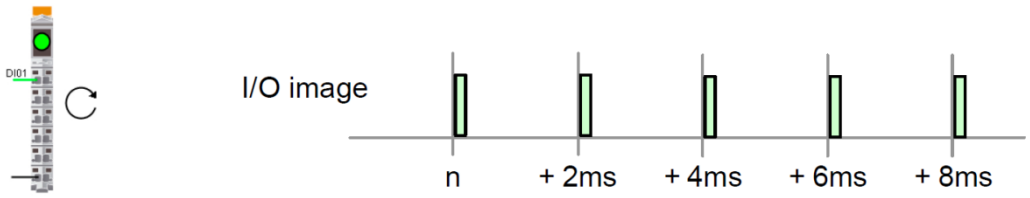


그림 47: 입력 이미지 읽어오기

어플리케이션에서 입력 데이터는 테스크 클래스가 시작될 때 설정된 사이클 타임보다 오래된 데이터가 들어오진 않는다; 출력 데이터는 적어도 그 시간이 지난 후에 이 이미지를 사용해서 쓰여진다.

테스크 클래스에서 I/O 데이터

I/O 스케줄러는 테스크 클래스에 I/O 데이터를 제공되거나 테스크 클래스가 시작될 때 제어한다.
기본적으로 I/O 스케줄러는 1000 us 의 시스템 틱(system tick)으로 설정되어 있다.
이 타이밍은 시스템 구성(system configuration)에 있는 “Timing” 카테고리에서 설정할 수 있다.

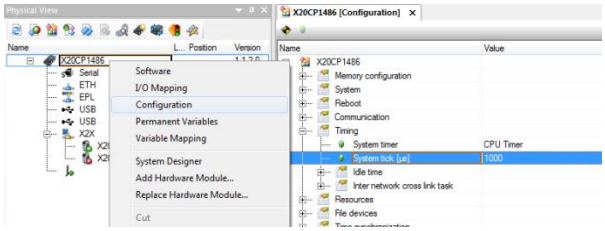


그림 48: 설정가능한 system tick

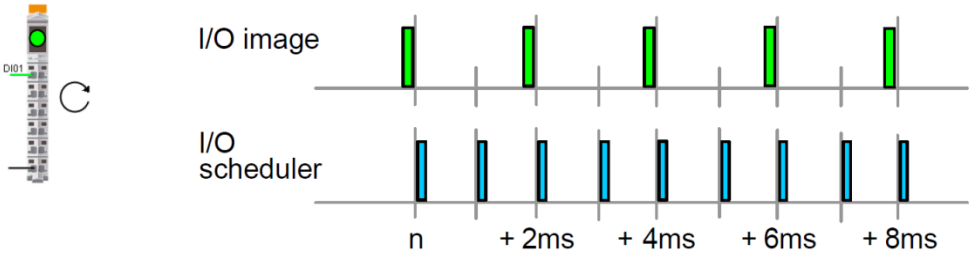


그림 49: 입력 이미지 읽어오기

이 경우에, I/O 스케줄러는 I/O 이미지 업데이트보다 두 배 빠르게 호출된다.

I/O 스케줄러는 정수 비율 동기로 필드버스 I/O 사이클을 동작시킨다. 타이머와 비율은 필요에 따라 설정할 수 있다.

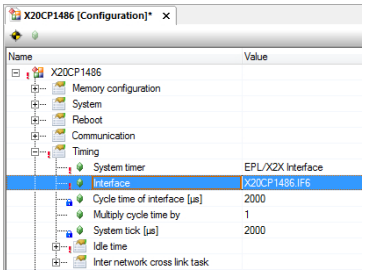


그림 50: 시스템 tick 동기화

1:1 비율을 가지는 필드버스 타이머(X2X 또는 POWERLINK) 설정은 I/O 스케줄러를 I/O 설정 사이클 타임마다 호출할 수 있다.

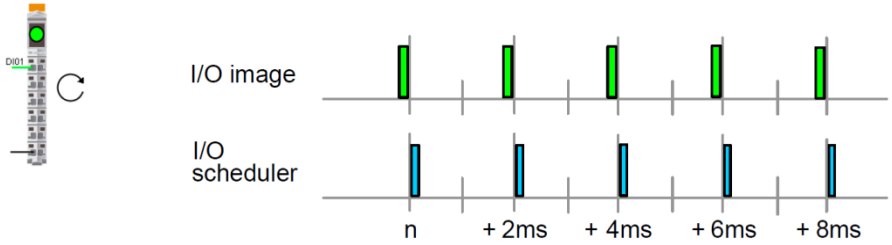


그림 51: 입력 이미지 읽어오기

I/O 관리는 이미지에서 변수로 I/O 데이터를 할당한다.

위의 예제에서 테스크 클래스의 테스크들은 I/O 스케줄러가 2 millisencods 마다 호출될 때의 결과를 보여준다:

테스크 클래스 #1

I/O 스케줄러는 10ms 마다 테스크 클래스 #1 이 시작한다. 그 테스크들은 할당된 변수에 대한 입력 이미지(초록색 화살표)와 내부 작업을 제공한다.

테스크 클래스 #1 안에 테스크 끝에서, 할당된 출력 변수들은 출력 이미지에 쓰여진다(빨간색 화살표).

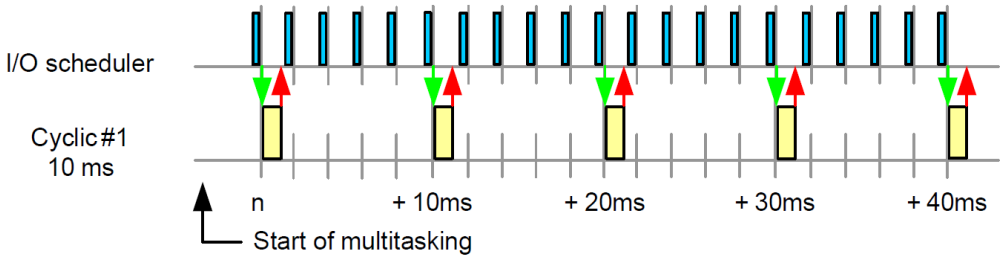


그림 52: 테스크 클래스 #1 에서 I/O 관리

테스크 클래스 #4

I/O 스케줄러는 n+50 / n+150 에서 테스크 클래스 #4 를 시작한다. 그 테스크는 할당된 변수들에 대한 입력 이미지와 내부 작업을 제공한다 (초록색 화살표).

테스크 클래스 #4 안에 테스크 끝에서, 할당된 출력 변수들은 출력 이미지에 쓰여진다(빨간색 화살표).

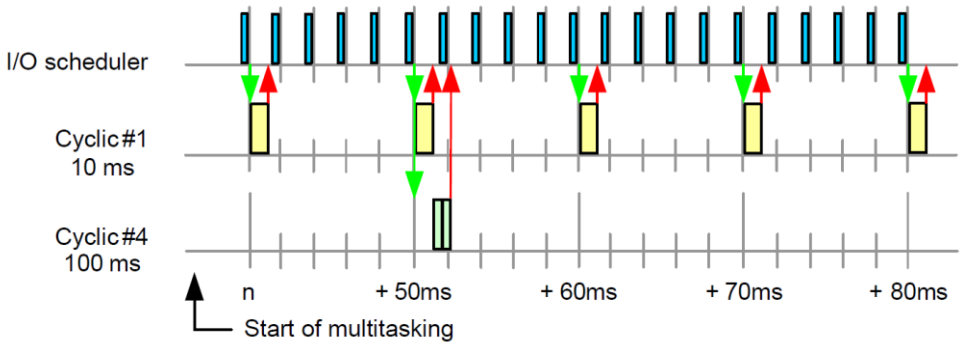


그림 53: 테스크 클래스 #4 에서 I/O 관리

Real-time operating system ₩ Target systems ₩ SG4 ₩ I/O management

4.3.7 사이클 타임과 허용오차(Cycle time 과 Tolerance)

테스크 클래스는 **사전에 정해진** 사이클 타임으로 설정된다. 테스크 클래스에 있는 모든 테스크는 설정된 사이클 타임 안에서 실행되야 한다.

프로젝트가 생성되거나 고객 요구사항에 따라 변경되었을 때 테스크 클래스의 사이클 타임은 수정된다.

만약 테스크 클래스 안에 테스크 들의 런타임 합이 설정된 사이클 타임을 초과한다면, 사이클 타임 위반(cycle time violation)이 발생한다. 사이클 타임이 초과될 때 서비스 모드로 부팅되는 것을 시스템에서 보호하도록 허용오차(Tolerance)을 설정할 수 있다.

사이클 타임과 허용오차는 테스크 클래스의 특성에서 설정할 수 있다. 테스크 클래스 단축메뉴에서 Properties(특성) 창을 연다.

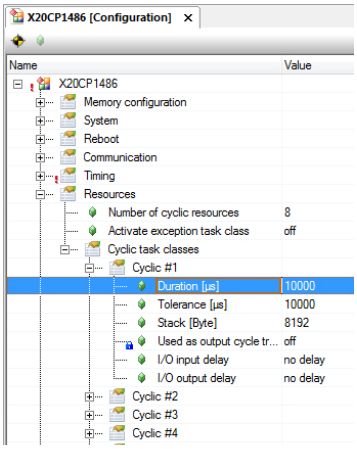


그림 54: 사이클 타임과 허용오차 설정하기

테스크 클래스 설정 런타임이 초과되면, 테스크 클래스의 다음 시작이 원래 설정된 사이클의 시작으로 옮겨진다. 이는 어플리케이션에 타이밍 문제를 발생시킬 수 있다.

Programming ₩ editors ₩ Configuration editors ₩ Hardware configuration ₩ CPU configuration

예제: 사이클 타임 위반(cycle time violation) 발생

테스크 클래스 #1 에서 작동하는 **"Loop"** 테스크에서 사이클 타임 위반을 발생시키기 위해서 Watch 창에 있는 **"udiEndValue"** 변수의 값을 변경하라.
테스크의 런타임 작동과 사용가능한 idle time 을 **Profiler** 로 확인하라. 사이클 타임 위반이 발생한 후, 원인분석을 위해 Automation Studio **Logger** 를 사용하라.

이 예제는 세 가지 과정으로 구성된다:
1) "udEndValue" 변수를 50000 로 설정
2) 단계적으로 "udEndValue" 변수 값을 증가
3) "udEndValue" 변수를 사이클 타임 위반이 발생할 때까지 증가

각 스텝마다 Profiler 결과를 분석해야 한다.

Step 1: "udEndValue" 변수 50000 으로 설정하기

- 첫 번째 단계에서, 변수 "udEndValue"의 값을 50000 로 설정

첫 번재 단계는 "udEndValue" 변수의 값을 50000 로 설정한다.

현재 실행 시간은 Profiler 를 사용해서 설정할 수 있다. 이 과정은 다음 단계에 설명되어 있다.

실행 시간을 결정하기 위한 **Profiler**

단축 메뉴 <Open> / <Profiler>를 선택해서 소프트웨어 configuration 으로부터 Profiler 를 연다.

이 과정을 수행하면서, “Loop” 테스크의 Watch window 를 확인 하는 것을 추천한다,
소프트웨어 구성과 Profiler 를 항상 열어둔다.
해당하는 workbook 을 선택해서 출력을 변환시키는 것도 가능하다.

그림 55: Profiler view 바꾸기

기본 설정으로, Automation Runtime 은 모든 런타임 작동을 기록한다.
이 과정 동안 유저 테스크, 테스크 클래스의 실행 시간과 예외적인 실행이 기록된다.
설정을 편집하기 위해서 Profiler 툴바에 “Stop” 아이콘을 사용해서 현재 기록 작업을 중지시켜야 한다.
설정 창을 열기 위해, Profiler 툴바에 “Configuration” 아이콘을 클릭하라.
다음 그림에 보이는 것처럼 설정하라:

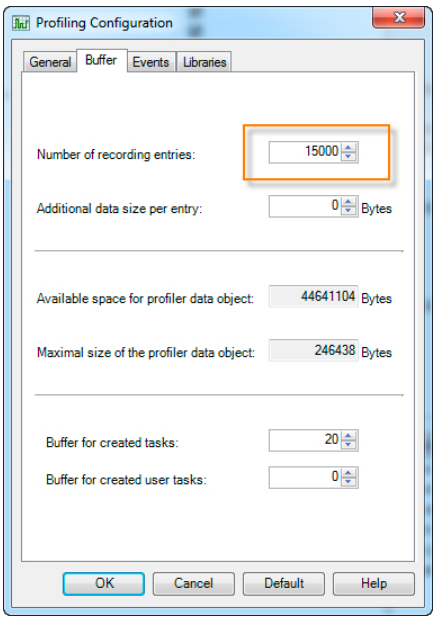


그림 56: Profiler 설정 1

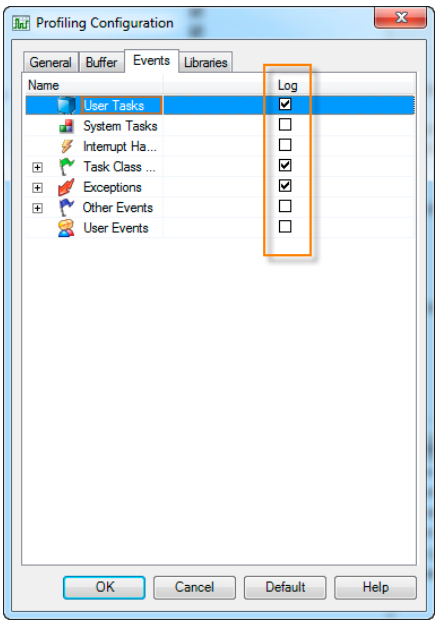


그림 57: Profiler 설정 2

Profiler 툴바에 “Install” 아이콘을 사용해서 타겟 시스템의 변경사항을 저장해라. 기록은 설정이 변경된 직후 다시 시작한다.

“Stop” 아이콘을 사용해서 기록 작업을 중단하고 Profilder 툴바에 “Upload data” 아이콘을 클릭하여 Automation Studio 에서 기록된 데이터를 불러올수 있으며, “Graph” 아이콘을 클릭하여 볼 수 있다.

시작 시점에 따라, 기록 작업은 다음과 같이 보일 수 있다:

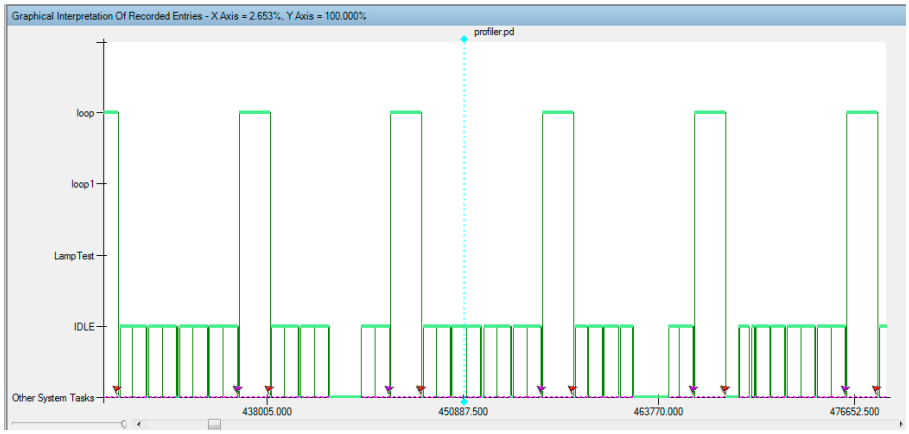


그림 58: 첫 번재 Profiiler 관리 결과

예제: Profiler 기록 확인하기

Profiler 기록의 결과를 Automation Studio 도움말 또는 “TM233 – Automation Studio Diagnostics” 교육 자료을 이용해서 평가해라.

Diagnostics and service ₩ diagnostics tools ₩ Profiler

Step 2: 단계적으로 “udEndValue” 변수 값 증가시키기

- 이번 단계에서는 Profiler 툴바에 “Start” 아이콘을 사용해서 Profiler 를 재시작하라
- “udEndValue” 값을 단계적으로 증가시켜라(e.g. 10,000).
- “Loop” 테스크 실행 시간을 Profiler 를 사용해서 관찰하라.

Profiler 툴바에 있는 아이콘을 이용해서 레퍼런스 커서(reference cursor)를 “Loop”의 시작에 두고 커서(curspor)를 “Loop” 끝에 두면, 정확한 실행시간을 알아낼수있다.

"Loop" 테스크는 10 ms(millisecond) 테스크 클래스에서 작동한다. 아래의 그림에 따라, 이미 실행 시간이 16.5 ms 에 도달했기 때문에 사이클 타임 위반이 발생된다. 허용오차 – 테스크 클래스 #1 의 특성에서 10ms 로 설정된 – 의 효과가 나타난다.

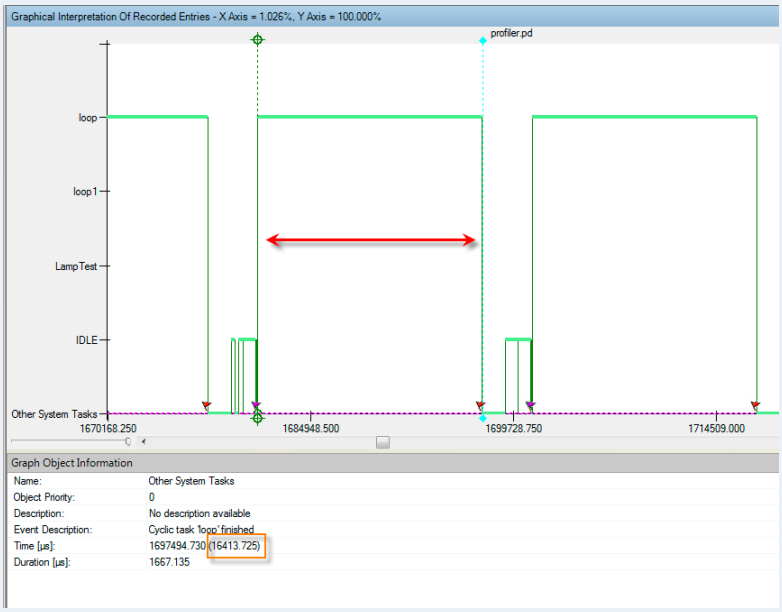


그림 59: 사이클 타임 초과, 허용 오차의 효과

Step 3: 사이클 타임 위반이 발생할 때까지 "udEndValue" 변수 증가시키기

- 변수 "udEndValue"를 사이클 타임 위반이 발생할 때까지 증가시켜라.

4.3.8 사이클 타임 위반(Cycle time violation)

Automation Runtime 이 사이클 타임 위반을 감지하면, 타겟 시스템은 서비스 모드로 부팅된다.

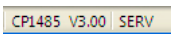


그림 60: 서비스 모드에서의 X20CP1485

예제에서, Watch 창에서 변수 변경으로 인하여 사이클 타임 위반이 발생한다. 사이클 타임 위반이 발생할 때, Watch 창에 보이는 모든 변수는 "frozen" 되고 서비스 모드로 재시작 된 후 0 으로 초기화 된다.

Diagnostics and service ₩ Diagnostics tools ₩ Logger window

Logger 는 **소프트웨어 구성**에서 **<Open> / <Logger>** 를 누르면 열 수 있다.

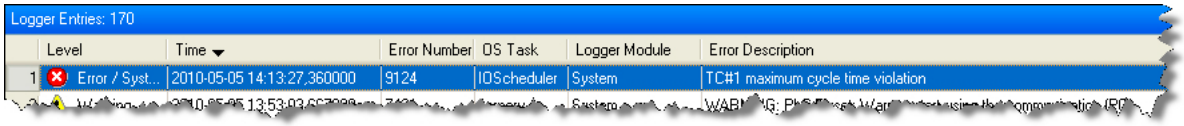


그림 61: Logger 에서 에러의 원인 분석

테스크 클래스 #1 에서의 사이클 타임 위반은 에러의 원인에 속한다.

사이클 타임 위반의 원인은 Profiler 를 사용해서 찾아낼 수 있다. 이를 위한 에제가 트레이닝 메뉴얼 TM233 – Automation Studio Diagnostics 에 포함되어 있다.

파워 OFF/ON 또는 Automation Studio 내의 재시작에 의해서 타겟 시스템이 재시작 되면, 이런 경우는 런모드(RUN)에서 부팅된다.

4.3.9 어플리케이션 프로그램을 사용하여 사이클 타임 위반에 대응하기

실시간 생산 시스템은 사이클 타임이 심하게 초과되었을 때 서비스 모드로 스위치되면 안된다.

Execption 프로그램을 사용하면 예외의 상황에 대응할 수 있다.

Exception 테스크 클래스는 Resources 카테고리의 CPU configuration 특성에서 설정가능하다.

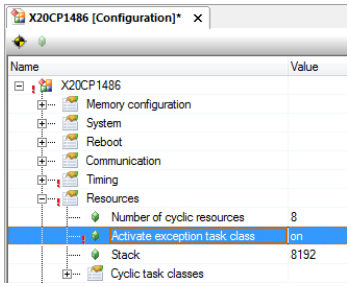


그림 62: exception 테스크 클래스의 활성화

Logical View 에 있는 프로그램은 소프트웨어 구성(software configuration)에서 exception 테스크 클래스에 추가할 수 있다.

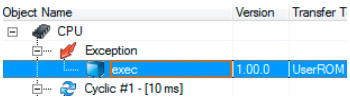


그림 63: exception 테스크 클래스 설정

Exception 테스크의 특성에 있는 exception 넘버는 어떤 exception 이 그 테스크를 작동시켰는가에 따라 결정된다.

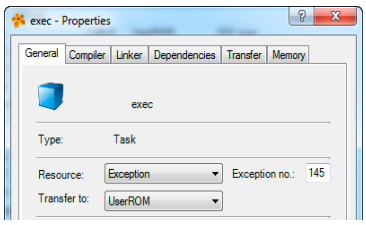


그림 64: exception 테스크 설정

런타임에 발생한 사이클 타임 위반(exception no.145)은 어떤 어플리케이션의 응답을 포함하는 테스크를 호출할 것이다.
발생가능한 다른 exception 들은 도움말을 참조하라.

Programming ₩ Editors ₩ configuration editors ₩ Hardware configuration ₩ CPU configuration ₩ Resources

Real-time operating system ₩ Target systems ₩ SG4 ₩ Runtime performance ₩ Exceptions

4.4 I/O 관리

컨트롤러의 주요 기능은 입력과 출력 상태를 결정론적으로 그리고 I/O 단자와 어플리케이션 사이로 가능한한 빠르게 이동시키는 것이다.

Automation Runtime 에서 I/O 관리는 **response time** 과 **configuratbility** 에 대한 높은 요구사항을 만족하도록 디자인된다.

I/O manager 는 테스크 클래스가 시작되기 전에 입력 이미지를 I/O 단자로 전달하고 테스크 클래스의 테스크 끝으로 출력 이미지를 전달한다.

테스크 클래스#1 은 테스크 클래스의 끝에 있는 출력으로부터 흔들림 없는 응답을 위해 설정 될 수 있다.

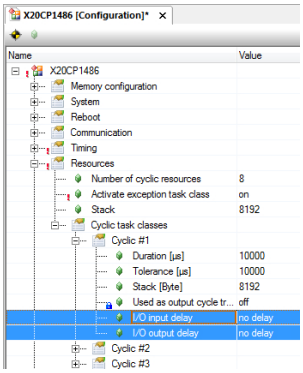


그림 65: 실행중인 출력 데이터 설정

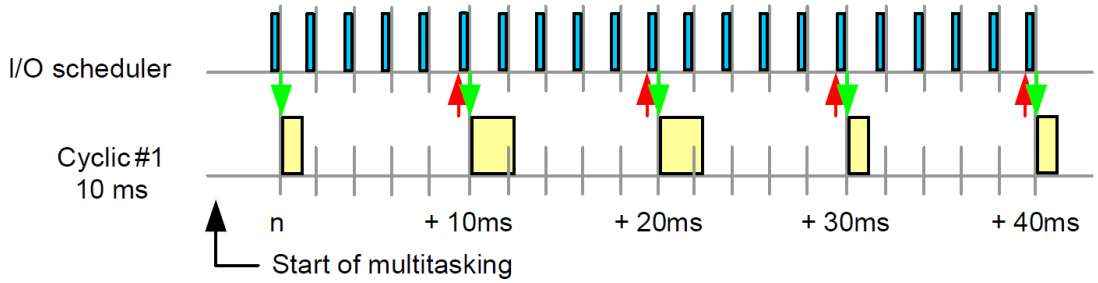


그림 66: 사이클 끝부분에서의 흔들림 없는 출력 이미지

I/O manager 는 I/O configuration 과 I/O 매핑에 의해 컨트롤된다.

Real-time operating system ₩ Target systems ₩ SG4 ₩ I/O management
Real-time operating system ₩ Target systems ₩ SG4 ₩ I/O management ₩ Method of operation

4.4.1 Startup

컨트롤러 부팅은 **활성화 I/O configuration** 을 I/O 모듈로 전달하고 인터페이스와 필드버스 장치를 초기화시킨다.
이는 유효한 I/O 데이터가 초기화 서브루틴에 접속할 수 있게 만들어 준다.

4.4.2 I/O 데이터 맵핑

I/O 단자로부터 차단된 입력 데이터들은 메모리에 저장된다. **I/O 맵핑**은 변수에 데이터를 맵핑하는데 사용된다.

출력 데이터는 차단된 출력 데이터를 사용해서 역순으로 쓰여진다.

다음에 오는 그림은 startup 동안 모듈에 전달되는 I/O configuration 과 일치하는 변수에 I/O 데이터를 할당시키는 I/O 맵핑 사이의 관계를 설명해준다.

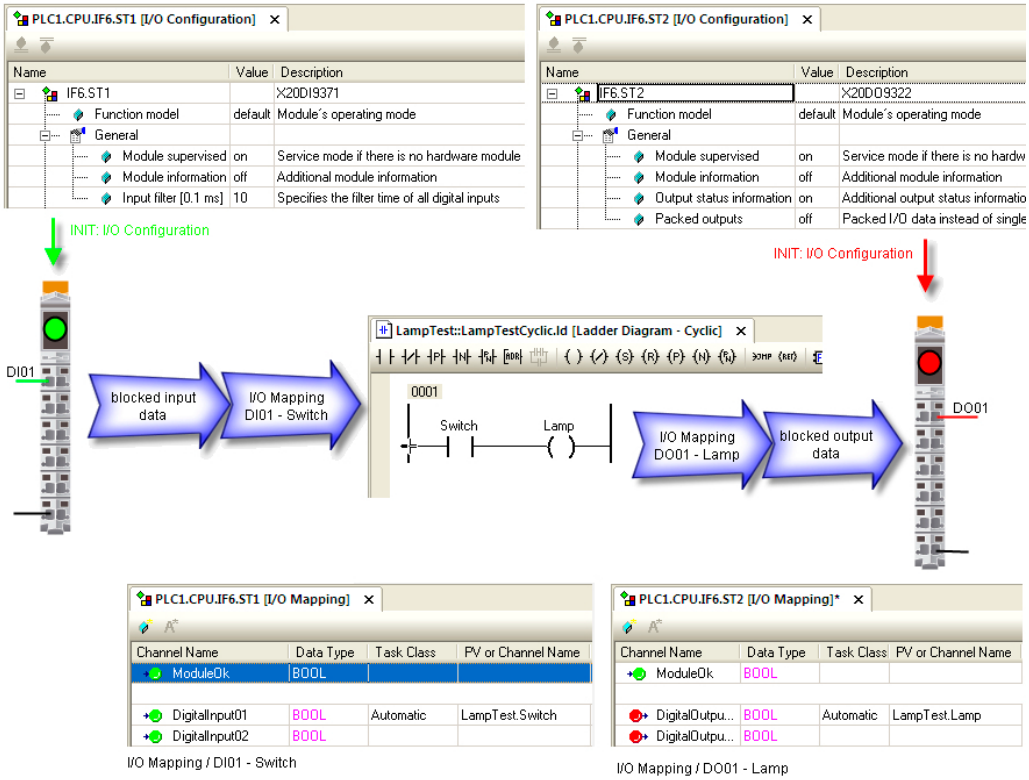


그림 67: 기본적인 I/O 기능

4.4.3 I/O 맵핑

I/O 맵핑은 어떤 I/O 데이터가 변수에 할당되는지를 결정한다.

프로그램에서 사용되는 .var 파일에 있는 각변수들은 그것의 scope 상관없이 I/O 모듈의 채널에 할당된다.

Physical View 에서 단축메뉴에 있는 **<I/O mapping>**을 선택하면 요구된 I/O 모듈에 변수들을 할당시킬 수 있다.

변수는 선택된 모듈에서 I/O 맵핑 에디터 안에 있는 각각의 채널에 할당된다.

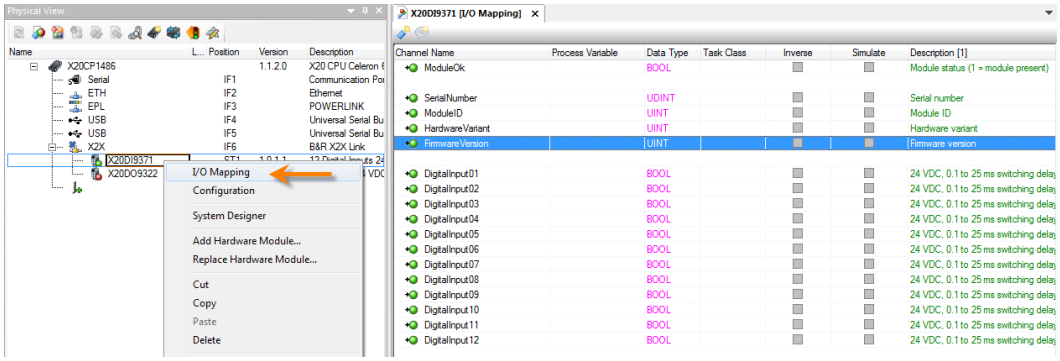


그림 68: 디지털 입력 카드의 I/O 맵핑

전역 변수에서, 채널은 I/O 이미지 데이터가 전달되야 하는 테스크 클래스에 할당되어진다.

만약 "Automatic"이 설정되고 프로젝트가 빌드되면, Automation Studio 는 자동으로 변수가 사용될 수 있는 가장 빠른 테스크 클래스를 선택할 것이다.

Programming ₩ Editors ₩ Configuration editors ₩ I/O mapping

4.4.4 I/O 설정

특정한 모듈 특성은 별다른 프로그래밍없이 I/O configuration 에서 설정될 수 있다.

원격 B&R I/O 모듈의 기능은 점점 더 많아지는 옵션과 모듈들이 사용될 수 있는 작동 모드를 가동시키기 위해 급격히 증가할 것이다.

Physical View 에서 단축메뉴에 있는 <Configuration>을 선택하면 요구된 I/O 모듈에 I/O 채널이 설정된다.

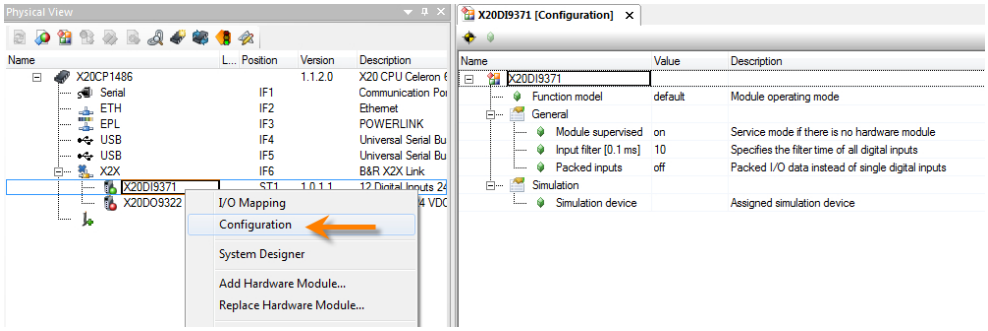


그림 69: 디지털 입력 모듈 설정

Programming ₩ Editors ₩ Configuration editors ₩ Hardware configuration ₩ I/O configuration

4.4.5 I/O 모듈의 에러 관리

I/O 모듈의 에러 관리의 주된 특징은 각각 설정된 I/O 모듈이 I/O 시스템에 의해 보여지는 것이다.

사용자는 시스템이 에러 상황에 어떻게 응답하는지 설정할 수 있다.

I/O configuration 안에 있는 "Module supervise" 특성을 이용해서 I/O 모듈의 모니터링을 enable (default stting) 또는 disable 할 수 있다.

모니터링 활성화

Automation Runtiem 에 의해 모듈이 활발하게 모니터링될 때, 아래와 같은 상태가 서비스모드로 변환될 수 있다:

- 슬롯에 결정된 모듈이 존재하지 않을 때 (not connected).
- 슬롯에 물리적으로 연결된 모듈이 슬롯에 실제로 설정된 모듈과 일치하지 않을 때
- I/O 시스템에서 모듈을 찾을 수 없을 때 (e.g. Module defective).

모니터링 비활성화

모니터링이 비활성화되면, "ModuleOK" 채널에 변수를 맵핑하는 것은 어플리케이션으로부터 발생할 수 있는 에러 상황에 반응하는 것이 가능하도록 만든다.

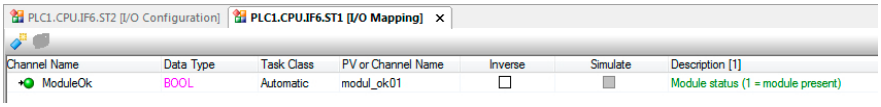


그림 70: 어플리케이션을 사용한 모듈 모니터링

Real-time operating system ₩ Target systems ₩ SG4 ₩ I/O management ₩ Method of operation ₩ Error handling

5 멀티태스킹 시스템의 상호작용

이번 챕터는 Automation Runtime 이 하나 또는 다수의 프로그램이 실행되고 변경될 때 어떻게 작동하는지에 대해 설명한다.

5.1 다른 테스크에 의해 중단되는 테스크

테스크 클래스 우선순위는 높은 우선순위의 테스크가 장시간 작동하는 낮은 우선순위의 테스크를 인터럽트 하는 것이 가능하게 한다.

예제: Profiler 를 사용하여 시스템에 있는 테스크 클래스를 설명하기

"Loop"와 **"Loop1"** 테스크를 사용해서, Watch 창을 열고 **"udiEndValue"** 의 마지막 값을 변경하면 **"Loop1"** 테스크는 **"Loop"** 테스크의 정확하게 **두 배**만큼 중단된다.

"Loop" 테스크에 **"udiEndValue"** 변수의 초기값을 2,000 으로 설정하라.

Profiler 측정은 다음과 같다:

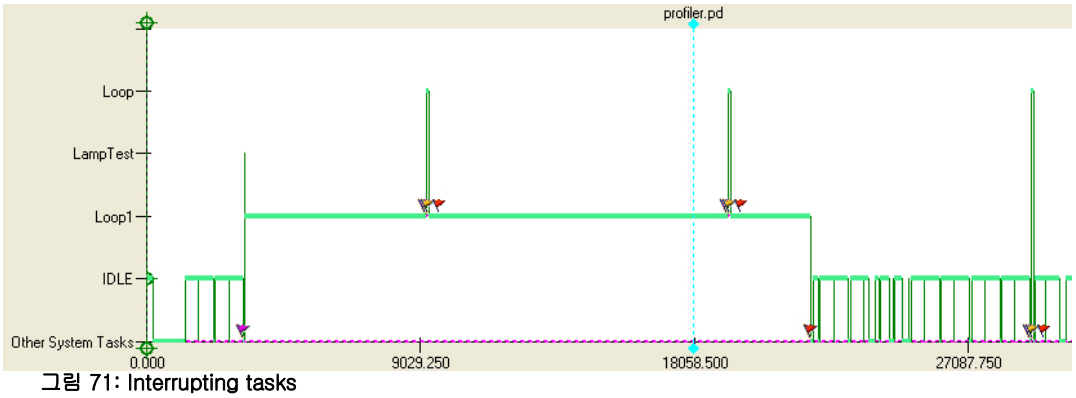


그림 71: Interrupting tasks

"Loop1" 테스크가 테스크 클래스 #4 에서 실행될 때, 입력 이미지는 테스크가 완벽하게 실행될 때까지 사용할 수 있다.

5.2 I/O 사이클과 테스크 클래스 동기화하기

자연스럽게, 테스크 클래스 #1 은 10 ms 로 설정한다.

그러나 POWERLINK 또는 X2X 를 사용한 빠른 스피드의 I/O 데이터 관리에 충분하지 않다.

4.3.6 “I/O 맵핑 다루기 – I/O 스케줄러”에서 설명했듯이, 데이터 검색은 필드버스 I/O 사이클과 동기화될 수 있다. 데이터가 I/O 검색 사이클에서 처리되도록 하기 위해, 테스크 클래스는 더욱 빠르고 더욱 자주 호출되야 한다.

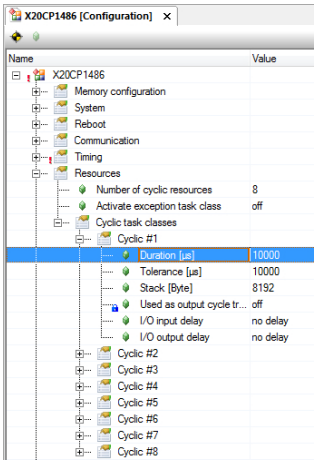


그림 72: 테스크 클래스 #1 의 사이클 타임 설정

Programming ₩ editors ₩ Configuration editors ₩ Hardware configuration ₩ CPU con-figuration

5.3 높은 허용오차(tolerance)을 가지는 테스크 클래스

우선순위가 낮지만 가능한 빠르게 처리되는 테스크는 테스크 클래스 #8 에 할당한다.

이 테스크 클래스의 디폴트(default) 사이클 타임(cycle time)은 10 ms 이고 허용 오차(tolerance)는 30 sec 이다.

테스크 클래스 #8 에 할당되야 하는 테스크:

- 어플리케이션 프로그램으로부터의 파일 접근
- 복잡하고 / 시간에 상관없는 계산들

5.4 프로그램 전달

프로그램 코드 또는 시스템 설정이 변경된 후, 프로그램을 타겟 시스템에 전달한다.

5.4.1 다운로드 모드

프로그램 전달과 관련해서, 꼭 고려해야할 두 가지가 있다:

- 개발동안 다운로드하기 – 실시간 생산 시스템에 효과가 없다.
- 런타임 동안 다운로드하기 – 실시간 생산 시스템에 효과가 있다.

사용자는 상황에 따라 적절한 다운로드 모드를 Configuration View 에서 선택해야 한다.

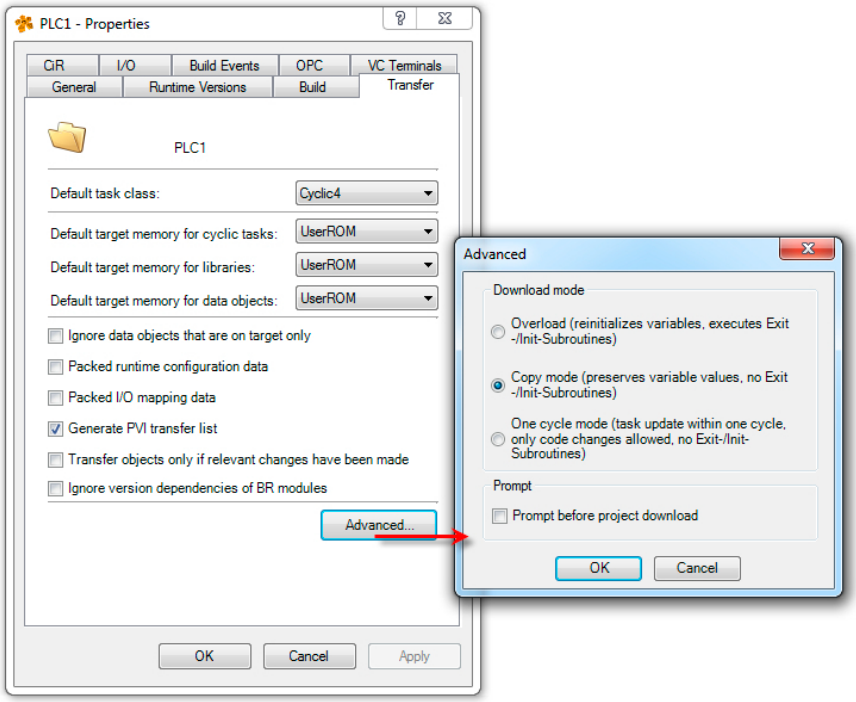


그림 73: Configuraiton View 에서 다운로드 모드 설정

개발 중 다운로드 – Overload mode

Overload mode 는 개발 도중 프로그램 코드를 자주 변경할 경우 사용해야 한다. 새로운 구성(new configuration)의 디폴트 모드이다.

Overload mode 는 실시간 생산 시스템에 적절하지 않다.

런타임 중 다운로드 – One Cycle mode

실시간 생산 시스템을 변경하는데 적합하다. 한 사이클당 하나의 프로그래만이 변경된다. 이는 실시간 생산 시스템의 가장 빠른 전환시간을 보장한다.

런타임 중 다운로드 – Copy mode

실시간 생산 시스템을 변경하는데 적합하다. 프로그램은 다중 사이클동안 전달된다.

Real-time operating system ₩ Target systems ₩ SG4 ₩ Download ₩ Copy mode

- Overload
- One cycle mode
- CopyMode

5.4.2 Exit 서브루틴

어떤 테스크의 Exit 서브루틴은 테스크가 삭제될 때 호출된다.
만약 리소스(메모리, 인터페이스)가 초기화 또는 사이클 서브루틴에서 사용된다면, 이러한 리소스들은 해제된다.

6 요약

운영 시스템은 각각의 타겟 시스템에서 일관적인 리소스 관리를 하면서 어플리케이션과 하드웨어 사이에 인터페이스를 제공한다.

멀티 태스킹은 어플리케이션을 모듈화 구성으로 디자인한다. 테스크에서 어플리케이션 준비는 최상의 리소스 사용을 가능하게 한다.

그림 74: Automation Runtime 타겟 시스템

멀티 태스킹 시스템에 접촉함으로써 어플리케이션의 타이밍은 사용자의 특별한 요구사항을 만족하도록 설정된다.

테스크는 조금 더 빠른 테스크 클래스에서 빠르게 실행되고 자주 실행되어야 하며, 시간에 따르지 않아도 되는 다른 중요한 테스크들은 조금 더 느린 테스크 클래스에서 실행하면 된다.

이는 리소스를 효율적으로 사용하는 동시에 어플리케이션과 기계의 성능을 최고로 끌어올릴 수 있는 최적의 설정을 지원한다.

7 부록

7.1 “Loop” 샘플 예제

“Loop” 프로그램

“Loop” 프로그램은 Structured Text 프로그래밍 언어로 만들었다. FOR 루프의 시작과 끝을 변수로 설정하여 값을 변경할 수 있다. 끝 값을 증가시킴으로써 CPU 로드를 증가시켜서 cycle time violation 을 발생 시킬 수 있다.

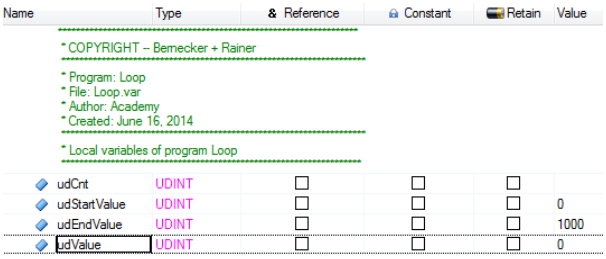


그림 75: “Loop” 변수 선언

```
PROGRAM _INIT
    udEndValue := 50;
END_PROGRAM

PROGRAM _CYCLIC

    (* implementation of program Loop *)
    FOR udCnt := udStartValue TO udEndValue DO
        udValue := udValue + 1;
    END_FOR

    gMain := gMain + 1;

END_PROGRAM
```

그림 76: Structured Text 로 “Loop”실행

Training modules
TM210 – Working with Automation Studio
TM213 – Automation Runtime
TM223 – Automation Studio Diagnostics
TM230 – Structured Software Development
TM240 – Ladder Diagram (LD)
TM241 – Function Block Diagram (FBD)
TM242 – Sequential Function Chart (SFC)
TM246 – Structured Text (ST)
TM250 – Memory Management and Data Storage
TM400 – Introduction to Motion Control
TM410 – Working with Integrated Motion Control
TM440 – Motion Control: Basic Functions
TM441 – Motion Control: Multi-axis Functions
TM450 – ACOPOS Control Concept and Adjustment
TM460 – Initial Commissioning of Motors
TM500 – Introduction to Integrated Safety
TM510 – Working with SafeDESIGNER
TM540 – Integrated Safe Motion Control
TM600 – Introduction to Visualization
TM610 – Working with Integrated Visualization
TM630 – Visualization Programming Guide
TM640 – Alarm System, Trends and Diagnostics
TM670 – Advanced Visual Components
TM800 – APROL System Concept
TM811 – APROL Runtime System
TM812 – APROL Operator Management
TM813 – APROL XML Queries and Audit Trail
TM830 – APROL Project Engineering
TM890 – The Basics of LINUX
TM920 – Diagnostics and service
TM923 – Diagnostics and Service with Automation Studio
TM950 – POWERLINK Configuration and Diagnostics

TM1010 – B&R CNC System (ARNC0)
TM1110 – Integrated Motion Control (Axis Groups)
TM1111 – Integrated Motion Control (Path Controlled Movements)
TM261 – Closed Loop Control with LOOPCONR
TM280 – Condition Monitoring for Vibration Measurement
TM480 – The Basics of Hydraulics
TM481 – Valve-based Hydraulic Drives
TM482 – Hydraulic Servo Pump Drives