



TM250 Memory Management and Data Storage

I 버전 정보

버전	날짜	수정내역	번역	검수
1.0	2017.10.20	첫번째 버전 TM250TRE.00 V1.0.1	정진강	임은

Table 1: Versions

선행 및 필요 조건

교육 자료	TM213 – Automation Runtime TM246 – Structured Text(ST)
소프트웨어	Automatio Runtime 4.21 Automation Studio 4.2
하드웨어	ArSim / X20CP1586

II 목차

1	소개	1
1.1	학습 목표.....	1
2	변수, 상수 그리고 데이터 타입	2
2.1	일반 정보.....	2
2.1.1	이진법과 십진법 시스템.....	3
2.1.2	변수와 상수 비교하기.....	6
2.1.3	변수, 상수, 데이터 타입 선언하기.....	7
2.2	기본데이터 타입.....	8
2.2.1	IEC 61131-3 기본 데이터 타입의 개요.....	8
2.2.2	데이터 타입 변환.....	9
2.3	배열, 구조체 그리고 열거형.....	10
2.3.1	배열.....	10
2.3.2	구성요소와 서브레인지.....	13
2.3.3	구조체 타입.....	14
2.3.4	열거형 데이터 타입(Enumerated data types).....	16
2.4	문자열(string).....	18
2.4.1	프로그램 코드에서 문자열 사용하기.....	18
2.4.2	문자열 평션.....	19
2.5	주소, 메모리 사이즈 그리고 동적 변수.....	22
3	메모리와 메모리 관리	24
3.1	변수 선언에서 메모리 초기화.....	24
3.2	프로그램 코드에서 메모리 복사와 초기화.....	25
4	라이브러리 사용하기	27
4.1	일반 정보.....	27
4.2	사용자 라이브러리 생성하기.....	31
4.3	B&R 기본 라이브러리 예제.....	35
5	데이터 처리의 기초	37
5.1	정렬.....	37
5.2	데이터 포맷.....	38
5.3	데이터 일관성.....	39
6	데이터 저장하기과 관리하기	40

6.1	메모리 블록 예약하기.....	40
6.2	Technology Guard 에 데이터 저장.....	41
6.3	데이터 오브젝트.....	42
6.4	파일 시스템에 파일 저장.....	44
6.5	Mapp technology 레시피 관리.....	45
6.6	데이터 베이스.....	47
7	데이터 전달과 커뮤니케이션.....	48
7.1	커뮤니케이션에 관한 일반 정보.....	48
7.2	커뮤니케이션 라이브러리.....	51
8	요약.....	53

1 소개

이 교육자료는 효율적으로 데이터를 포맷, 관리, 구성할 수 있도록 돕는다. 가장 중요한 측면 중 하나는 사용자 정의 데이터 타입을 프로그래밍, 데이터 저장, 커뮤니케이션에서 올바르게 사용하는 것이다. 변수, 데이터 타입, 상수를 정확하게 사용하는 것은 에러 방지에 도움이 될 뿐만 아니라, 전반적인 어플리케이션의 유연성과 일관성을 향상시킨다.



그림 1 메모리와 변수



그림 2 데이터 처리와 저장



그림 3 커뮤니케이션과 데이터 변환

여기 있는 정보는 데이터를 처리하고 결정하는데 어떤 툴과 어떤 절차가 필요한지를 결정하는 것을 도와준다.

다양한 테스트를 해결을 위한 의사 결정 프로세스를 위하여, 저장 가능한 데이터 포맷과 저장 위치에 대한 전반적인 내용을 제공한다. Automation Studio 라이브러리 예제는 라이브러리 평션을 빠르게 테스트하기 위한 방법을 제공한다. 데이터는 쉽게 파일에 저장되거나 네트워크를 통해 전달된다. Automation Studio 는 고객 맞춤형 라이브러리를 생성할 수 있고, 이를 위한 광범위한 도움말을 제공한다.

모든 예제 프로그램은 Structured Text 프로그래밍 언어를 기반으로 개발되어 있다. IEC text 포맷을 기반으로 변수, 데이터 타입, 상수 선언에 사용된다.

1.1 학습 목표

이 교육자료는 메모리 관리와 데이터 저장에 대한 기초를 세우기 위해 도움을 줄 어플리케이션 예시와 예제를 사용한다.

- 변수와 데이터 타입을 초기화하고 사용하는 방법을 배울 것이다.
- 간단한 데이터 타입과 복잡한 데이터 타입을 사용하는 방법을 배울 것이다.
- 열거형과 사용자 데이터 타입을 사용하는 방법을 배울 것이다.
- 어플리케이션에서 메모리를 관리하는 다양한 옵션에 대해 배울 것이다.
- B&R 기초 라이브러리에서 평션와 평션 블록을 정확하게 호출하는 법을 배울 것이다.
- 사용자 라이브러리를 생성하는 법을 배울 것이다.
- 데이터 처리에 관한 기초를 배울 것이다.
- 데이터 저장, 파일 관리 그리고 메모리 블록 관리 기초에 대해 배울 것이다.
- B&R 기초 라이브러리를 사용한 데이터 전달과 커뮤니케이션에 이용할 수 있는 옵션에 대해 배울 것이다.
- B&R 라이브러리 예제 불러오기와 사용법 그리고 통합 Automation Studio 도움말 사용법에 대해 배울 것이다.

2 변수, 상수 그리고 데이터 타입

고정된 메모리 주소는 과거 프로그래밍의 것이다.; 요즘 프로그래밍은 특정한 이름을 가지는 상징적 요소를 사용한다. 이 요소를 변수(variables)라고 부른다.

기본 데이터 타입은 변수 값 범위를 결정할 뿐만 아니라 메모리에서 그 변수가 어느 정도의 공간이 필요한지를 결정한다. 또한 데이터 타입은 값의 부호가 있는지 없는지, 십진법인지, 텍스트인지, 날짜인지, 시간인지까지 규정한다.

IEC text 포맷을 사용한 이 교육 자료에서 변수와 데이터 타입 선언이 설명되어 있다. Automation Studio 사용자는 테이블 에디터를 열 것인지, 텍스트 에디터를 열 것인지 선택할 수 있다. 선언 파일을 더블 클릭하면, 테이블 에디터가 자동으로 열린다.

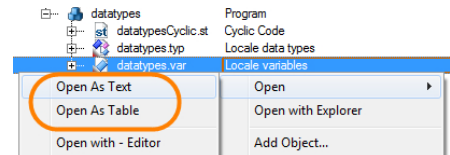


그림 4 텍스트 또는 테이블로 선언창 열기

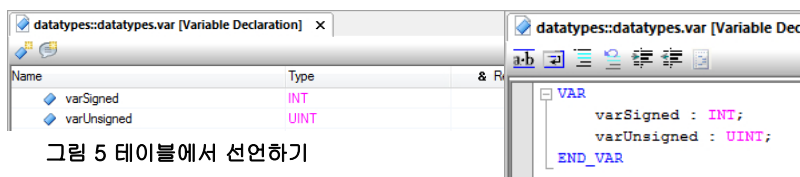


그림 5 테이블에서 선언하기

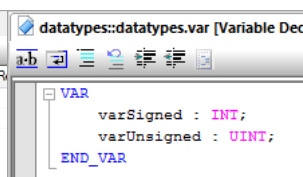


그림 6 텍스트에서 선언하기

Programming W Editors W Table editors W Declaration editors

2.1 일반 정보

가장 기본 레벨에서, 어느 컴퓨터든지 0 과 1 의 전자적 상태에 대한 해석과 그룹화를 통한 값 계산과 표시가 가능하다. 이러한 기본 기능을 잘 알고 있으면 유용하다.

당신이 이 시스템을 이해한다면, 시스템 오류를 쉽게 찾아 내고 프로그래밍을 할 수 있다.

2.1.1 이진법과 십진법 시스템

Bit 는 정보의 가장 작은 단위이고 0 과 1 로 이루어져 있다. IEC¹에 따르면, bit 는 BOOL 과 관련있다.

다른 데이터 타입은 8 개로 나뉜 다수의 bit 로 구성된다. 다음으로 큰 단위는 byte 이다. 1byte 는 8bit 로 구성된다.

byte 안의 bit 는 오른쪽에서 왼쪽으로 쓰여진다, Bit 0 에서부터 가장 큰 값인 Bit 7 까지. Bit 2(실제로는 3 번째에 있는 bit)는 십진수 4 의 값을 가진다.

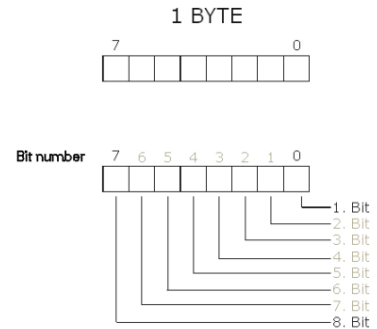


그림 7 byte 의 이진수 표현

1byte 는 두 부분으로 나뉜다. 반은 nibble 로 분류된다. 논리적으로, 더 낮은쪽 nibble 은 low nibble 이라고 부르고, 높은 쪽 nibble 은 high nibble 이라고 부른다.

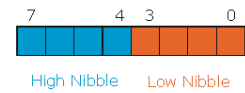


그림 8 High nibble 과 low nibble

가장 큰 bit 는 high nibble 에서 찾을 수 있다.

Byte 안에 있는 각각의 bit 는 0 과 1 의 값을 가진다. 1byte 는 256 개로 이루어진 0 에서 255 값을 가진다.

Bit pattern	Bit number	Decimal value	2 ^{bit number}
00000001	0	1	2 ⁰
00000010	1	2	2 ¹
00000100	2	4	2 ²
00001000	3	8	2 ³
00010000	4	16	2 ⁴
00100000	5	32	2 ⁵
01000000	6	64	2 ⁶
10000000	7	128	2 ⁷

표 1 Byte 에서 각 bit 값

두 이진수를 더하는 예제이다.

+	00000001 One
	00000001 One
=	00000010 Two (십진수 10 이 아니다!)

표 2 올림이 있는 이진수의 덧셈

다음에 적용해 보자:

0 + 0 = 0

0 + 1 = 1

1 + 1 = 0 + 다음 bit 로 올림 됨

¹ IEC 61131-3 규범에는 데이터 타입, 프로그래밍 언어 그리고 특별한 플랫폼에 얽매이지 않는 파일 구성방식 등이 명시되어 있다.

이진 시스템에서 음수가 있다면, 가장 높은 bit 의 부호가 반대로 바뀐다. 그 결과로, 7bit 가 값을 표현한다.

Bit 7 은 부호이다	
x0000000	0 부터 6 까지의 bit 는 수의 범위
x1111111	가장 큰 양의 십진수 127

표 3 음의 정수 값에 대한 bit 패턴

음수는 2 의 보수를 이용해서 합해진다. 양의 십진수로부터 이 bit 패턴이 생성된다. 이 bit 패턴은 거꾸로 바꾼다음 1 을 더한다. 그 결과는 아래와 같은 음수 bit 패턴으로 나타난다.

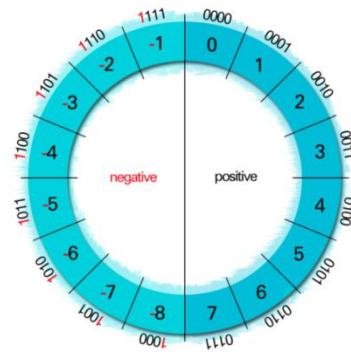


그림 9 이진 시스템에서 음수 표현

이진 시스템에서 음수 표현

	0000011	십진수 “3”
	1111100	Bit 를 거꾸로
+	0000001	1 을 더함
=	1111101	십진수 “-3”

표 4 2 의 보수를 이용한 계산

정확한 숫자의 표현은 데이터 타입에 따른다.

부호를 가지는 데이터 타입(SINT, INT, DINT)에 음의 값이 부호를 가지지 않는 데이터 타입(USINT, UINT, UDINT)에 할당된다면, bit 패턴은 똑같은 것이다. 하지만 그 값은 다르게 나타난다.

이 예제는 데이터와 데이터 타입간의 관계를 설명한다.

선언	<pre> VAR varUnsigned : USINT := 0; varSigned : SINT := 0; END_VAR </pre>
프로그램 코드	<pre> varSigned := -22; (*bit pattern 1110 1010 *) varUnsigned := varSigned; </pre>

표 5 signed data type 에 unsigned data type 할당하기

“varUnsigned” 변수는 십진수 234 로 보여진다. 이는 bit 패턴 1110 1010 을 따른다.

Bit 패턴은 변하지 않는다. 그러나 데이터 타입이 변하면(unsigned) 다른 값이 보인다.

Name	Type	Value
varSigned	SINT	-22
varUnsigned	USINT	234

그림 10 십진수로 표현된 두 변수의 값

Name	Type	Value
varSigned	SINT	2#1110_1010
varUnsigned	USINT	2#1110_1010

그림 11 이진수로 표현된 두 변수의 값



Programming W Variables and data types W Data types W Primitive data types W InT (INTERGER)

Programming W Variables and data types W Variables W Bit addressing

10 진법과 다르게, 16 진법은 16 개의 값 (0 에서 F 까지)을 가진다.

0000	십진수 0	16 진수 0
0001	십진수 1	16 진수 1
0010	십진수 2	16 진수 2
0011	십진수 3	16 진수 3
0100	십진수 4	16 진수 4
0101	십진수 5	16 진수 5
0110	십진수 6	16 진수 6
0111	십진수 7	16 진수 7
1000	십진수 8	16 진수 8
1001	십진수 9	16 진수 9
1010	십진수 10	16 진수 A
1011	십진수 11	16 진수 B
1100	십진수 12	16 진수 C
1101	십진수 13	16 진수 D
1110	십진수 14	16 진수 E
1111	십진수 15	16 진수 F

표 6 이진수의 16 진수 표현



Nibble 과 16 진수

0100	1011	75 에 해당함		(64+8+2+1=75)
0100		High nibble	=	4
	1011	Low nibble	=	B
0100	1011	Both nibbles	=	16#4B=75

표 7 2 진수를 16 진수로 변환

Nibble 은 2 진수에서 16 진수로 간단히 복사되고 각각 따로 쓰인다.



2 진수와 16 진수의 메모리 크기

2 진수				16 진수	메모리 크기
00000000				16#00	1 byte
00000000	00000000			16#0000	2 bytes
00000000	00000000	00000000	00000000	16#00000000	4 bytes

표 8 이진수와 16 진수의 메모리 크기와 표현



16 진수 표현은 주로 로거 목록 또는 주소를 보여줄 때 사용된다.
Signed 와 unsigned 변수에서 에러 위치 검출은 bit 패턴을 비교하면 효율적으로 찾을 수 있다.

Watch 창은 2 진법, 10 진법, 16 진법 변수 값을 보여주는 옵션이 있다.

IEC 규범에서는 문자 그대로 2#0000_1001 을 2 진수로 나타내고 16 진수로는 16#09 로 나타낸다.



Programming W Standards W Literals in IEC languages

2.1.2 변수와 상수 비교하기

변경될 수 있고 런타임에 새로운 값을 넣을 수 있는 메모리에 변수가 저장되어 있다. 몇가지 변수에 관한 예제들은 디지털과 아날로그 입/출력 뿐만 아니라 보조 플래그도 포함한다.

변수와 다르게, 상수는 선언되며 값이 할당된다. 이 값은 런타임동안 더 이상 변하지 않는다. 상수는 제한된 값 또는 범위 제한 같은 것을 포함한다.

기본 데이터 타입들은 어떤 경우에도 사용가능하다. “IEC 6113-3 의 기본 데이터 타입의 개요”

예제: 상수 값 할당하기

- 1) 상수 선언
MAX_INDEX 변수를 declaration 창에 생성하라. 데이터 타입은 USINT.
“Constant” 옵션을 선택하고 값은 123 할당하라.
- 2) 상수에 값 할당하기
프로그램 코드에서, 상수에 새로운 값 43 을 할당하라.
- 3) 컴파일러로부터 출력 확인
프로그램을 컴파일하고 컴파일러로부터 출력을 분석하라.



상수에 값을 할당하는 것은 프로그램 코드 안에서는 불가능하다. 컴파일러가 다음과 같은 에러 메시지를 나타낼 것이다.:

Error 1138: cannot write to variable 'MAX_INDEX'



Programming W Variables and data types W Variables W Constants

상수는 배열을 초기화하는데 사용할 수 없다. 그러나, 프로젝트 셋팅 “Allow extension of IEC standards”를 사용하면 할 수 있다.



Project management W The workspace W General project settings W Settings for IEC compliance

2.1.3 변수, 상수, 데이터 타입 선언하기

이 교육 자료는 Automation Studio 사용자 인터페이스에 대해 자세히 설명하지 않을 것이다.

옆의 이미지는 글로벌, 로컬 선언 파일이 있는 Logical View 를 보여준다.

변수와 데이터 타입은 테이블 에디터와 텍스트 에디터에서 설정된다.

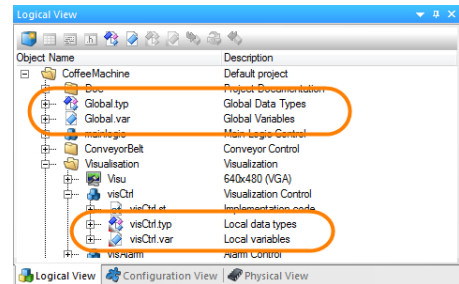


그림 12 Logical view 에서 선언파일

변수 선언 창

변수와 상수는 일반적으로 .var 확장자를 가지는 파일에 저장된다. 프로그램 이름과 변수 선언 파일은 새로운 프로그램이 프로젝트에 삽입될 때 생성된다.

데이터 타입 선언 창

사용자에 의한, 열거형(enumerated) 그리고 파생(derived) 데이터 타입은 항상 .typ 확장자를 가지는 파일에 저장된다. 새로운 프로그램이 프로젝트에 삽입될 때, 사용자는 항상 데이터 타입 선언 파일도 생성한다.



Programming W Editors W table editors W declaration editors W Variable declaration
Programming W Editors W table editors W declaration editors W Data type declaration
Programming W Editors W General operation W smart Edit

초기화

변수, 상수, 데이터 타입 그리고 구조체 변수는 declaration 에디터에서 직접 초기화 된다.



Programming W Editors W table editors W declaration editors W Variable declaration
Programming W Editors W table editors W declaration editors W Data type declaration
Programming W Editors W General operation W Dialog boxes for input support W dialog box for initializing structure types and instances

관찰(Scope)

선언된 변수와 데이터 타입의 관찰은 Logical View 에서 선언한 파일 위치에 따른다.



Programming W Variables and data types W variables W Scope of declarations

2.2 기본데이터 타입

초기 데이터 타입은 모든 파생 데이터 타입의 기본 형태이다.

2.2.1 IEC 61131-3 기본 데이터 타입의 개요

다음에 오는 목록은 IEC61131-3 규범에 있는 모든 기본 데이터 타입과 그 범위를 포함한다.

2 진법 / Bit series	Signed inter- gers	Unsigned integers	Floating point	String	Time, date
BOOL	SINT	USINT	REAL	STRING	TIME
BYTE	INT	UINT	LREAL	WSTRING	DATE_AND_TIME (DT)
WORD	DINT	UDINT			DATE
DWORD					TIME_OF_DAY (TOD)

표 9 IEC 데이터 타입 개요

초기 데이터 타입의 완성된 목록, 이 들의 사용 범위와 값 범위는 Automation Studio 도움말에서 확인 할 수 있다.



데이터 타입 BYTE, WORD, DWORD 는 순수한 bit 배열이다. 산술 연산자는 컴파일러로부터 허용되지 않는다.



IEC 데이터 타입의 한가지 특징은 사용하고 있는 플랫폼과 독립적이라는 것이다. 이는 IEC 데이터 타입이 프로세서 양식이나 프로그램 코드에 관계 없이 항상 같은 값의 범위를 가진다는 것을 의미한다.



Programming W Variables and data types W Data types

2.2.2 데이터 타입 변환

프로그래밍 동안, 데이터 타입을 다른 것으로 변환하는 것이 필요할 때가 있다. 작은 범위를 가지는 데이터 타입의 변수를 큰 값을 가지는 데이터 타입에 할당할 때, 암시적 데이터 변환이 일어난다. 그 반대가 되면 (큰 범위가 작은 범위로), 사용자는 프로그램 코드에서 변화를 명확하게 관리해야 한다.

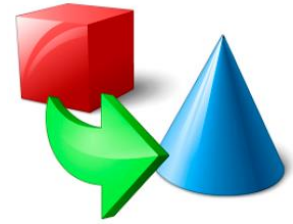



그림 13 데이터 타입 변환

암시적 데이터 변환(Implicit data type conversion)

암시적 데이터 변환은 컴파일러가 하나의 데이터 타입을 다른 것으로 변환할 때 발생한다.

 이 프로그램 코드에서, 범위가 큰 데이터 타입에 작은 범위를 가지는 데이터타입을 할당된다.


선언	<pre>VAR bigValue : DINT := 0; smallValue : SINT := 0; END_VAR</pre>
프로그램 코드	<pre>bigValue := smallValue;</pre>

표 10 암시적 데이터 변환

이런 종류의 할당은 값이 새로운 데이터 타입 안에서 충분한 공간을 가질 수 있도록 해준다. 사용자는 직접 변환 하지 않아도 된다. 암시적으로 컴파일러가 변환을 실행한다.

명시적 데이터 타입 변환(Explicit data type conversion)

범위가 큰 데이터 타입의 값이 범위가 작은 데이터 타입에 할당되면, 사용자가 직접 변환을 실행해야 한다.

 선언

선언	<pre>VAR bigValue : DINT := 0; smallValue : SINT := 0; END_VAR</pre>
프로그램 코드	<pre>smallValue := bigValue;</pre>


표 11 필요에 의한 명시적 전환


이 경우에, 컴파일러는 다음의 메시지를 출력한다:
Error 1140: Incompatible data types: Cannot convert DINT to SINT.

“AslecCon”라이브러리에 있는 평션들은 변환하는데 사용한다. 이 라이브러리는 자동으로 Automation Studio 프로젝트가 생성될 때 포함된다.

위의 예제의 변환은 아래와 같이 수행해야 한다.

변환을 위한 값은 변환 평션 괄호 안에 있다.

 선언	<pre>VAR bigValue : DINT := 0; smallValue : SINT := 0; END_VAR</pre>
프로그램 코드 표 12 명시적 변환 실행	<pre>smallValue := DINT_TO_SINT(bigValue);</pre>


 Programming W Libraries W IEC 61131-3 functions W AslecCon W Function blocks and functions

2.3 배열, 구조체 그리고 열거형

사용자가 정의한 데이터 타입은 다른 기초 데이터 타입을 기반으로 생성된다. 이 방법을 컴퍼지션이라고 부른다. 사용자가 정의한 데이터 타입은 기본 데이터 타입으로 구성되어 있다.

이렇게 파생된 데이터 타입들은 다음과 같다:

- 배열과 다차원 배열
- 구조체
- 직접 파생된 타입과 서브레인지
- 열거형(Enumerators)

 Programming W Variables and data types W Data types W Derived datat types

2.3.1 배열

원시적인 데이터 타입과 달리, 배열은 같은 데이터 타입을 갖는 여러 변수들이 복합되어 있다. 각 개별 요소들은 배열의 이름과 인덱스 번호를 이용해서 접근할 수 있다.

배열에 접근하기 위한 인덱스는 실제 배열 크기를 넘어서 접근하면 안된다. 배열의 크기는 변수로 선언할때 정의되어 있다.

프로그램에서, 인덱스는 고정 숫자, 변수, 상수, 열거형(enumerated element)로 쓸 수 있다.

Name	Typ	Wert
aPressure	INT[0..9]	
aPressure[0]	INT	123
aPressure[1]	INT	555
aPressure[2]	INT	0
aPressure[3]	INT	552
aPressure[4]	INT	32767
aPressure[5]	INT	9700
aPressure[6]	INT	0
aPressure[7]	INT	9
aPressure[8]	INT	0
aPressure[9]	INT	13

그림 14 0 에서 9 범위이면서, 서로 다른 10 개 배열 요소를 가지는 INT 데이터 타입 배열

배열 선언과 사용

배열을 선언할 때, 데이터 타입과 치수를 설정해야한다. 통상적인 관습상 가장 작은 인덱스는 0이다. 배열 사이즈가 10 이라면 최대 인덱스는 9이다.



	선언	<pre>VAR aPressure : ARRAY[0..9] OF INT := [10 (0)]; END_VAR</pre>
	프로그램 코드	<pre>(* Assigning the value 123 to index 0 *) aPressure[0] := 123;</pre>

표 13 크기 10인 배열 선언, 시작 인덱스 = 0

만약 인덱스 10을 가지는 배열에 접촉하려 한다면, 다음과 같은 에러 메시지가 뜰 것이다:

프로그램 코드	<pre>aPressure[10] := 75;</pre>
에러 메시지	<pre>Error 1230: The constant value '10' is not in range '0..9'.</pre>

표 14 범위 바깥의 배열에 접근했을 때

 10개 요소를 가진 배열 인덱스를 선언하고자 한다면, 선언 에디터에 “USINT[0..9]” 또는 “USINT[10]”을 입력한다. 두 경우에, 배열의 시작 인덱스는 0이고 최대 인덱스는 9로 배열이 생성될 것이다.

예제: “aPressure” 배열 생성

- 1) 로지컬 뷰에 “int_array” 프로그램 추가
- 2) 변수 선언 창 열기
- 3) “aPressure” 배열 선언
배열은 10 요소를 포함해야한다. 가장 작은 인덱스는 0. 데이터 타입은 INT 형을 사용한다.
- 4) 프로그램 코드에서 배열 사용
프로그램 코드에서 배열 접근을 위해 인덱스를 사용한다. 고정된 숫자, 상수, 변수를 이용한다.
- 5) 프로그램 코드에서 가능하지 않는 배열 접근
배열의 인덱스 10을 접근하고 메시지 윈도우 창을 확인하라.

aPressure[10] := 123 같이 할당할 경우, 컴파일러는 아래와 같은 에러 메시지를 보고할 것이다.



Error 1230: The constant value '10' is not in range '0..9'.

변수를 사용한다면 컴파일러는 배열 접근을 확인하지 못할 수 있다.

```
Index := 10;
aPressure[index] := 123;
```

상수(Constants)를 이용한 배열 선언

선언과 프로그램에서 사용된 고정된 숫자 값은 프로그래밍을 다루기 어렵고 관리하기 힘들다. 더 좋은 아이디어로 숫자 상수를 사용한다.

배열의 하한, 상한 값을 상수로 정의한다. 이 상수들은 프로그램 코드에서 사용할 수 있고 배열 인덱스 제한을 둘 수 있다.


 선언	<pre> VAR CONSTANT MAX_INDEX : USINT := 9; END_VAR VAR aPressure : ARRAY[0..MAX_INDEX] OF INT ; index : USINT := 0; END_VAR </pre>
프로그램 코드	<pre> IF index > MAX_INDEX THEN Index := MAX_INDEX; END_IF aPressure[index] := 75; </pre>

표 15 상수를 사용한 배열 선언



프로그램 코드는 업데이트 되었고 배열 접근을 위한 인덱스는 배열 최대 인덱스 제안이 걸려있다. 장점은 배열 리사이징(크거나, 작거나)이 필요할 때 프로그램 코드 수정 없이 적용할 수 있다.



Programming W Variables and data types W Data types W Derived datat types W Arrays

예제 : 합과 평균 값 계산하기

“aPressure” 배열의 평균 값을 구하라. 프로그램은 배열 사이즈가 수정이 되더라도 이를 프로그램에서 다룰 수 있도록 프로그램 구조 설정하라.

- 1) 루프를 사용하여 합 계산
고정 숫자 값은 프로그램 코드에서 사용하지 않는다.
- 2) 평균 값을 계산
평균 값 변수의 데이터 타입은 배열의 데이터 타입(INT)과 같은 것으로 설정하라



배열 변수에서 배열 사이즈를 설정하기 위해 상수가 사용되었고 이는 루프 끝 값으로 사용할 수 있다. 평균을 구할 때, 같은 상수로 나누면 된다. 개별 배열 요소를 더할 때에는 반드시 출력 데이터 타입은 큰것을 사용해야 한다(예 DINT). 최종 값을 명확한 데이터 타입 변환기를 사용하여 INT 데이터 타입으로 변환시켜야 한다.

다차원 배열

배열은 여러개 차원으로 구성할 수 있다. 선언과 사용은 아래를 참조하라.


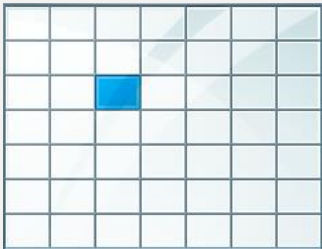


 선언	<pre>VAR Array2Dim : ARRAY [0..6,0..6] OF INT; END_VAR</pre>	
<hr/> 프로그램 코드	<pre>(*Counting starting with 0 *) Arry2Dim[2,2] := 11;</pre>	 <p>그림 15 3 행 3 열의 값</p>

표 16 7x7 의 2 차 배열의 선언과 접근

	<p>프로그램 코드상에서 고정 숫자, 상수 또는 열거형(enumerated element)를 이용해서 유효하지 않는 배열을 접근하려 한다면 컴파일러에 의해 감시되거나 예방된다.</p> <p>프로그램 코드상에서 변수를 이용해서 유효하지 않는 배열을 접근하려 한다면 컴파일러에 의해서 감시되지 않고 런타임 메모리 에러를 유발할 수 있다. 런타임 에러는 유효 범위 내에서 배열 인덱스를 접근하므로써 피할 수 있다.</p>
---	--

IEC Check 라이브러리를 Automation Studio 프로젝트에 추가하여 런타임 에러 위치를 찾는 데 도움을 준다.

	<p>Programming W Libraries W IEC Check library</p>
---	--

2.3.2 구성요소와 서브레인지

배열 뿐만 아니라 다른 파생 데이터 타입들도 기초 데이터 타입으로부터 파생된다.

기본 데이터 타입으로부터 직접적으로 파생 데이터 타입들을 생성하는 것은 가능하다. 기본 데이터 타입과 같은 특징을 가지는 새로운 이름의 새로운 데이터 타입을 만들어라.

새로운 데이터 타입에 초기값을 입력하라. 그 결과로, 이러한 데이터 타입의 모든 변수는 설정된 값을 가진다.

파생 데이터 타입에서 값 범위를 지정해야 한다. 이렇게 하면 값 범위를 설정할 필요없이, 이 데이터 타입의 변수에 값을 할당할 수 있다.



 서브레인지 변수	VAR varSubRange : USINT(24..48); END_VAR
서브 레인지 데이터 타입	TYPE Voltage_typ : USINT(12..24); END_TYPE

표 17 서브 레인지에서 변수와 데이터 타입 선언

 Programming W Variables and data types W Data types W Derived data types W Directly derived data types
Programming W Variables and data types W Data types W Derived data types W Subranges

예제: 서브레인지의 직접 파생된 타입 선언하기

새로운 데이터 타입 “pressure_typ”를 구성하라. 기본 데이터 타입은 INT 로 하라. 값의 유효 범위는 6500 과 29000 사이이다. 상수를 사용해서 이 값의 범위를 결정하라.

- 1) 직접 파생된 타입 선언
데이터 타입 선언 에디터를 열고 새로운 타입을 “pressure_typ”이름으로 선언하고 기본 데이터 타입을 “INT”로 할당하라.
- 2) 상수 선언
변수 선언 에디터를 열고, 상수 “MIN_VAL”과 “MAX_VAL”를 생성하고 위에 주어진 값을 할당하라.
- 3) 상수를 사용해서 “pressure_typ”의 서브레인지 결정
두 상수를 사용해서 새로운 데이터 타입에서 값의 범위를 설정해라.
- 4) 결과 테스트
변수 선언 에디터에서 데이터 타입 “pressure_typ”의 새로운 변수를 선언하라. 그 다음에 프로그램 코드에서 유효범위 바깥의 값을 할당하라.

2.3.3 구조체 타입

사용자 정의 데이터 타입으로 알려진, 구조체 는 기본 데이터 타입 또는 분할된 이름을 사용하는 구조체 그룹이다. 각각의 개별적인 요소들은 이름을 가지고 있다.

구조체는 기본적으로 서로 관계 있는 데이터와 값들을 그룹화하는데 사용된다. 항상 같은 재료를 사용하지만 다른 양으로 제조하는 레시피를 예로 들 수 있다.

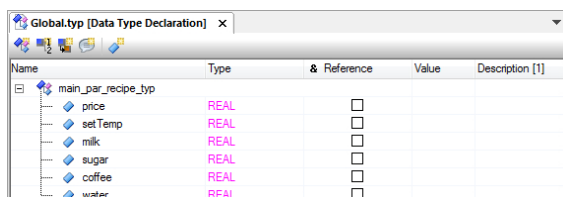


그림 16 Automation Studio 에서 구조체 선언

	구조체 선언	<pre> TYPE main_par_recipe_typ : STRUCT price : REAL; setTemp : REAL; milk : REAL; sugar : REAL; coffee : REAL; water : REAL; END_STRUCT; END_TYPE </pre>
	변수 선언	<pre> VAR AnyCoffee : main_par_recipe_typ; END_VAR </pre>
	프로그램 코드에서 사용	<pre> (*price for AnyCoffee*) AnyCoffee.price := 1.69; </pre>

표 18 구조체 선언과 프로그램 코드에서 사용

Programming W Variables and data types W Data types W Derived data types W Structures
 Programming W Editors W Table editors W Declaration editors W Data type declaration

예제: 구조체 “recipe_typ” 선언하기

“recipe_typ”의 이름으로 구조체를 선언하라.

이 구조체는 다음과 같은 요소들을 포함한다:

- price
- milk
- sugar
- coffee
- water

어느 데이터 타입이든지 각각의 요소들을 선택할 수 있다; 이는 프로그램에서 요소들이 어떻게 사용되는가에 따른다.

1) 구조체 선언

변수, 상수 그리고 데이터 타입


선언 에디터를 열고 “Add structure type” 아이콘에서 새로운 구조체를 생성하라. 이름을 “recipe_typ”로 할당하라. 위의 리스트에 있는 요소들을 추가하라.

2) 프로그램 코드에서 요소들을 초기화하라.

새로운 데이터 타입을 사용해서 이름이 “cappuccino”인 새로운 변수를 선언하라. 당신의 프로그램 코드에 이 변수를 사용하고 요소들을 초기화하라.

구조체 배열

구조체는 배열로 선언할 수 있다. 이 경우에 기본 데이터 타입의 배열과 같은 규칙이 적용된다. 다시 한번 언급하자면, 인덱스는 접속에 사용되고 메모리가 벗어나지 않도록 제한되어야 한다.

 선언	VAR aCoffee : ARRAY[0..5] OF recipe_typ; END_VAR
프로그램 코드	aCoffee[0].water := 12;

예제: “sugar”의 값을 동시에 추가하기

“recipe_typ” 구조체의 배열을 선언하라. 이 배열은 10 개 차수를 가진다. 배열을 임의의 값으로 초기화하라.

고정된 값 대신에 상수를 사용하라. “sugar”를 모든 구조체에 추가하고 당신의 제품에 평균적으로 얼마나 sugar가 필요한지 결정하라.

- 1) “aCoffee” 변수 선언
“recipe_typ” 데이터 타입을 사용하라. 이 배열 크기는 10이다.
- 2) 구조체 변수 초기화
구조체 변수는 소스코드에서 쓰여질 수도 있고 선언 에디터에서 쓰여질 수도 있다. 구조체 개별 요소들의 초기값은 데이터 타입 선언에서 설정할 수 있다. 초기값들은 변수 선언에서 중복 기재될 수도 있다. 또한 에디터를 사용해서 배열의 요소들이 이미 같은 값으로 초기화 되어있을 때 값 목록들을 복사할 수 있다.
- 3) “sugar” 값을 동시 입력
루프를 사용해서 “sugar”값을 입력할 수 있다. 상수를 사용해서 루프의 제한값을 설정하라.
- 4) “sugar”의 평균 값 계산
값을 입력한 후, “sugar”의 평균 값을 계산하라. 배열의 처음 인덱스와 마지막 인덱스 값의 변화를 항상 다르게 테스트하라. 이는 당신의 프로그램이 올바르게 동작하는지 확인할 수 있는 쉬운 방법이다.

2.3.4 열거형 데이터 타입(Enumerated data types)

열거형은 기본적으로 값 리스트를 열거하는 방법이다. 그러므로 열거형 데이터 타입은 변수 사용이 가능하다. 나열된 요소의 값 대신에, 변수는 알맞는 텍스트로 대체된다.

열거된 요소 값은 자동으로 0 부터 오름차순으로 정리된다.

열거형 타입과 그 안에 있는 요소들을 사용하는 것은 기계의 다른 상태를 나타내는데 유용하다.

	열거형 타입 선언	<pre> TYPE Color : (red, yellow, green); </pre> <p>END_TYPE</p>		그림 17 열거형 타입 선언							
변수 선언	<pre> VAR stepColor : Color; result : USINT; </pre> <p>END_VAR</p>	프로그램 코드	<pre> CASE stepColor OF red: result := 1; yellow: result := 2; green: result := 3; </pre> <p>END_CASE</p>	<table border="1"> <thead> <tr> <th>Name</th> <th>Typ</th> <th>Wert</th> </tr> </thead> <tbody> <tr> <td>stepColor</td> <td>Color</td> <td>red</td> </tr> </tbody> </table>	Name	Typ	Wert	stepColor	Color	red	그림 18 변수 모니터에서 열거형 타입
Name	Typ	Wert									
stepColor	Color	red									

표 19 열거형 타입 선언과 프로그램 코드에서 사용



Programming W Variables and data types W Data types W Derived data types W Enumeration

2.4 문자열(string)

문자열은 각 바이트의 순차적인 배치를 따른다. 각 바이트는 글자와 숫자로 이뤄진 character 또는 컨트를 character 를 포함한다. 함께 입력할 때, 이 character 열은 문자열로 구성된다.

만약 문자열이 10 개의 character 로 정의된다면, 10 개의 character 를 사용할 수 있다. 모든 문자열은 이진수 “0”(사용가능한 character 로 여기지 않는다)이 나오면 사라진다. 다시 말해, 10 개의 character 로 이루어진 문자열은 사실 11 바이트의 메모리를 차지한다.

사용가능한 character 의 한 부분이 실제로 사용가능하다면, 0 다음의 메모리는 정의되지 않고 남아있다.

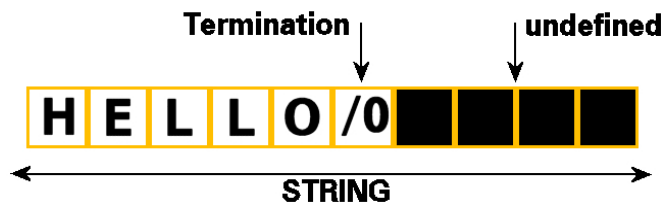


그림 19 character 10 개의 문자열, “Hello” 다음에 0 이 있다.


2.4.1 프로그램 코드에서 문자열 사용하기

아래 표는 선언 (초기화를 포함하여) 이 어떻게 다루어지는지를 보여준다. 사용가능한 character 의 개수는 대괄호 안에 명시된다. 선언 또는 프로그램 코드에서 문자열이 텍스트로 취급되는 것을 허용한다.

	선언	<pre>VAR sDescription : STRING[80] := 'Test'; END_VAR</pre>
	프로그램 코드	<pre>sDescription := 'Hello World'</pre>

표 20 문자열 선언, 프로그램 코드에서 사용, 텍스트 할당

문자열이 너무 긴 문자열 변수(선언된 문자열 변수보다 길게)에 할당된다면, 문자열은 컴파일러에 의해 길이가 줄어든다.



Programming W Variables and data types W Data types W Primitive data types W STRING, WSTRING

Programming W Standards W Literals in IEC languages

Programming W Standards W ASCII tables

예제 : 문자열 변수 사용하기

“recipe_typ”라는 문자열 변수를 선언하라. 10 개의 캐릭터 변수를 사용해야한다. 그 다음 “Automation Studio” 글자를 프로그램에서 그 변수에 할당시키고 변수 모니터를 확인하라.

- 1) “sDescription” 문자열 변수 선언
- 2) 프로그램에 선언한 문자열 변수 사용

```
“Automatino Studio” 글자 할당
sDescription := 'Automation Studio';
```

- 3) 변수 모니터 결과 확인
 - 변수 모니터에서 글자를 겹쳐 쓴 다음 어떤 일이 일어나는지 확인하라.

2.4.2 문자열 평션

문자열은 프로그램코드에서 몇가지 다른 방법으로 사용된다.

다음 기능을 지원한다:

- 문자열 비교
- 문자열로 변환
- 문자열을 변환
- 문자열 취급

대부분의 문자열 평션은 라이브러리 평션을 필요로 한다.

문자열 비교

문자열이 서로 동일한지 확인한다. 그 결과는 같거나 다르다이다.




 선언	<pre>VAR sTextA : STRING[80] := 'Perfection'; sTextB : STRING[80] := ' in Automation'; equal : BOOL; END_VAR</pre>
프로그램 코드	<pre>IF sTextA = sTextB THEN equal := TRUE; ELSE equal := FALSE; END_IF</pre>

표 21 문자열 비교

두 문자열이 일치하지 않는다면, 결과는 “equal := FALSE;” 이다.

 몇가지 프로그래밍 언어는 문자열이 대입 연산자에 할당되거나 IF 문장을 사용해서 숫자 변수같은것과 비교되는 것을 제공하지 않는다.

 두 문자열의 차이점을 자세히 보고 싶다면, “AsBrStr” 라이브러리의 “brsstrcmp” 평션을 사용하라.



Programming W Libraries W Configuration, system information, runtime control W AsBrStr
 Programming W Libraries W Configuration, system information, runtime control W AsBrWStr

문자열을 효율적으로 다루기 위해서는 평션들이 필요하다. 다음 단원에서는 사용가능한 라이브러리에 대한 개요와 이들을 어디에 사용할 수 있는지 설명한다.

문자열 관리 평션이 들어 있는 라이브러리:

- Stndard
- AslecCon
- AsBrStr
- AsBrWStr
- AsString

문자열 변환

서식이 정확하다면, 문자열 내용을 숫자로 변환할 수 있다. 그 반대도 역시 가능하다. 이 변환 평션은 AslecCon 라이브러리의 수많은 평션들 중 두 가지가 있다. AslecCon 라이브러리는 자동으로 새로운 Automation Studio 프로젝트에 추가된다.



“sPressure” 변수 내용을 숫자 값 “Pressure”로 변경한다.

선언	<pre>VAR Pressure : REAL; sPressure : STRING[80] := '12.34'; END_VAR</pre>
프로그램 코드	<pre>Pressure := STRING_TO_REAL(sPressure);</pre>

표 22 문자열을 Real 타입으로 변환

반대 변환은 다음과 같다.

선언	<pre>VAR Pressure : REAL := 12.34; sPressure : STRING[80]; END_VAR</pre>
프로그램 코드	<pre>sPressure := REAL_TO_STRING(Pressure);</pre>

표 23 Real 타입을 문자열로 변환



Programming W Libraries W IEC 61131-3 functions W AslecCon

문자열 처리

문자열은 다른 문자열과 비교하거나 숫자로 변환될 수 있을 뿐만 아니라, 문자열을 서로 연결할 수 있고(연속), 한 부분만으로 문자열을 검색할 수 있으며, 텍스트를 대체하거나 다른 문자열의 특정 위치에 삽입 할 수 있다.

이러한 업무는 “STANDARD” 라이브러리의 평션들을 사용해서 처리할 수 있다.



선언

```

VAR
sSourceString : STRING[80] := 'Strings in AS';
sFindString : STRING[80] := 'in';
position : INT;
END_VAR
프로그램 코드
position := FIND(sSourceString, sFindString);
    
```

표 24 다른 문자열 안에 문자열 위치 알아내기

문자열 4 번째 위치에서 처음으로 발견되었기 때문에 결과 값은 4 이다.



Programming W Libraries W IEC 61131-3 functions W STANDARD W Function blocks and functions W STRING handling functions

예제: 문자열 첨부

두 개 문자열에 각각 접근하라. “STANDARD” 라이브러리의 “CONCAT” 평션을 사용하라.

- 1) 변수 선언
초기값이 “Hello”인 “sText1” 변수, 초기값이 “World!”인 “sText2” 변수, sResult 변수를 선언하라.
- 2) “CONCAT” 평션 호출
- 3) 변수 모니터에서 결과 확인

추가 정보

“STANDARD” 라이브러리 평션들은 문자열 처리에 사용된다. 이 평션들은 항상 변수 데이터 길이를 확인하고 메모리 초과 발생을 막는다.

그러나, waudtlehls 데이터들은 일반적으로 32KB 로 제한된다. 보다 많은 데이터 양을 처리하기 위한 문자열 기능은 “AsBrStr” 라이브러리와 “AsBrWStr” 라이브러리에 있다.



“AsBrStr” 라이브러리에서 평션을 호출할 때, 타겟 변수에 결과 문자열이 충분한 공간을 가지고 있다면, 시스템이 확인하지 않는 것을 알아야한다. 만약 프로그래밍 에러가 있다면, 메모리 초과가 발생할 수도 있다. 이런 이유 때문에, 타겟 변수가 예약된 메모리 문자열을 수용하기에 충분한지 사전에 어플리케이션을 확인하는 것이 중요하다.

문자열의 마지막 길이는 “AsBrStr”라이브러리의 “brsstrlen” 평션 또는 “STANDARD” 라이브러리의 “LEN”를 사용하여 알 수 있다.



Programming W Libraries W Configuration, system information, runtime control W AsBrStr
Programming W Libraries W Configuration, system information, runtime control W AsBrWStr


2.5 주소, 메모리 사이즈 그리고 동적 변수


주소

변수 선언 창에 생성된 모든 변수, 상수, 배열, 구조체는 컴파일러로부터 메모리 주소가 할당된다. 이 주소는 데이터 컨트롤러의 시작 메모리를 표시한다.

런타임에서 이 메모리 주소는 더이상 변하지 않는다; 그러므로 이 변수들은 “static variables”로 참조된다. 변수가 위치한 주소는 ADR()평션을 사용해서 알아낼 수 있다.

변수의 메모리 주소는 특히 데이터를 평선나 평선 블록에 전달할 때 중요하다. 이 때, 데이터 시작 주소는 데이터 프로세싱 시작에 따라 전달된다. “라이브러리 사용하기”

	선언	<pre>VAR aCoffee : ARRAY[0..5] OF recipe_type; adr_index_0 : UDINT; END_VAR</pre>
	프로그램 코드	<pre>adr_index_0 := ADR(aCoffee[0]);</pre>
표 25 0 요소의 주소 검출		

 메모리 주소가 검출될 수 있지만, 프로그램에 고정된 값으로 사용되어야만 하는 것은 아니다. 이는 새로운 주소가 작동중인 시스템에 컨트롤러가 부팅될 때마다 할당되기 때문이다.


예제: 주소 알아내기

ADR() 평션을 사용해서 “aCoffee” 배열의 메모리 주소를 알아내라. 인덱스 0 과 인덱스 1 의 주소를 알아내야한다. 두 주소의 차이를 계산하고 기대했던 결과가 나왔는지 확인하라.

메모리 크기

고정된 변수는 메모리를 차지한다. 이는 각 변수의 데이터 타입에 따른다. 얼마나 많은 메모리가 필요한지 정확하게 아는 것이 필요할 수도 있다. 메모리 크기는 sizeof() 평션을 사용해서 알 수 있다.

배열에서, 메모리 총량은 배열에 사용된 데이터 타입의 합이다. 배열 요소들의 숫자 또한 계산된다.

	선언	<pre>VAR aCoffee : ARRAY[0..5] OF recipe_type; size_complete : UINT; size_single : UINT; num_elements : UINT; END_VAR</pre>
	프로그램 코드	<pre>size_complete := SIZEOF(aCoffee); size_single := SIZEOF(aCoffee[0]); num_elements := size_complete / size_single;</pre>
표 26 배열에서 요구되는 메모리와 그 요소들, 요소 숫자들		

예제: 메모리 크기 알아내기

“aCoffee” 배열 전체에 사용되는 메모리 양을 알아내라; 또한 인덱스 0의 배열요소 크기를 알아내라. 배열 요소의 숫자를 계산하라.



Programming W Libraries W IEC 61131-3 functions W OPERATOR W Function blocks and functions W Address and length functions W ADR
 Programming W Libraries W IEC 61131-3 functions W OPERATOR W Function blocks and functions W Address and length functions W SIZEOF

동적 변수

동적 변수는 변수 선언장에서 “REFERENCE TO” 키워드를 사용해서 선언할 수 있다. 컴파일러는 동적 변수에 개별적 메모리 주소를 할당할 수 없다. “ACCESS” 키워드를 사용하면 프로그램 상의 어떤 메모리 주소라도 접속할 수 있다. 접속 후에, 동적 변수는 언급된 메모리를 나타낸다. 이 데이터는 읽고 덮어쓰기가 가능하다.

예를 들어, “index” 변수를 어떤 메모리 주소가 접속 될 것인지 변화시키는데 사용할 수 있다. 변수 “dynRecipe”는 “index” 변수로부터 선택된 메모리 위치를 나타낸다.



선언

```
VAR
    aCoffee : ARRAY[0..5] OF recipe_type;
    dynRecipe : REFERENCE TO recipe_type;
    index : UINT;
END_VAR
```

프로그램 코드

```
dynRecipe ACCESS ADR(aCoffee[index]);
```

표 27 동적 변수의 선언, 메모리 주소에 접속



Programming W Variables and data types W Variables W Dynamic variables

3 메모리와 메모리 관리

응용 프로그램을 설계할 때, 시스템이 어떻게 동작할 것인지 이해해야 한다. 변수, 상수, 배열, 구조체가 초기화되는 방법 및 부팅하는 동안 어떤 일이 일어나는지 아는 것은 중요하다.

3.1 변수 선언에서 메모리 초기화

변수와 상수는 선언될 때 초기화된다. 변수에 초기 값이 할당되지 않는다면, 컨트롤러가 부팅될 때 항상 “0” 값을 받는다. 상수는 반드시 초기값이 할당되어야 한다.

배열과 구조체의 초기값은 변수 선언 편집창에서 할당된다. 게다가, 구조체의 개별 요소들은 데이터 타입이 선언될 때 초기값이 할당된다.

Name	Type	Constant	Retain	Value	Description [1]
gConveyor	conveyor_typ	<input type="checkbox"/>	<input checked="" type="checkbox"/>	(0)	info structure of conveyor
gBrewing	brewing_typ	<input type="checkbox"/>	<input type="checkbox"/>	(0)	info structure of brewing assembly
doDoseMilk	BOOL	<input type="checkbox"/>	<input type="checkbox"/>	FALSE	doser: milk
gFeeder	feeder_typ	<input type="checkbox"/>	<input checked="" type="checkbox"/>	(0)	info structure of feeder
axConveyor	ACP104XIS_typ	<input type="checkbox"/>	<input type="checkbox"/>		axis information of conveyor
doCupPull	BOOL	<input type="checkbox"/>	<input type="checkbox"/>	FALSE	pulls cup out of storage
gMainLogic	main_typ	<input type="checkbox"/>	<input type="checkbox"/>		info structure of main logic
axFeeder	ACP104XIS_typ	<input type="checkbox"/>	<input type="checkbox"/>		axis information of feeder
afWaterTemp	INT	<input type="checkbox"/>	<input type="checkbox"/>	0	actual temperature of water
diStartCoffee	BOOL	<input type="checkbox"/>	<input type="checkbox"/>	FALSE	start making coffee
doDoseSugar	BOOL	<input type="checkbox"/>	<input type="checkbox"/>	FALSE	doser: sugar
aoHeating	INT	<input type="checkbox"/>	<input type="checkbox"/>	0	heating control
doDoseCoffee	BOOL	<input type="checkbox"/>	<input type="checkbox"/>	FALSE	doser: coffee
gHeating	heating_typ	<input type="checkbox"/>	<input type="checkbox"/>	(0)	info structure of heating
doPumpWater	BOOL	<input type="checkbox"/>	<input type="checkbox"/>	FALSE	water pump
AnyCoffee	main_par_receipe...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	{price=-12.0,setTemp=-...	

그림 20 변수, 상수, 배열, 구조체가 있는 변수 선언 창

모든 변수와 상수는 컨트롤러의 DRAM 에 저장된다. 예를 들어서 배열 인덱스 범위 바깥에 접근하는, 정확하게 프로그래밍 되지 않는다면, 상수 값을 처리하는 것도 가능하다.

변수 값이 컨트롤러 재시작 후에도 남아있어야 한다면, “RETAIN” 옵션을 선언창에서 선택하면 된다. 그러면 파워가 꺼지거나 재시작 할 때에는 데이터가 컨트롤러 배터리 버퍼 SRAM 에 저장된다. 이에 대한 상세한 정보는 “TM213 – Automation Runtime”에 있다.



응용프로그램을 만들 때, 파워가 꺼지거나 재시작 할 때 변수가 메모리에서 어떻게 되는지 고려해야 한다.
응용프로그램은 어떤 초기화나 재시작 후에도 정확한 파라미터를 가지고 부팅되어야 한다.



Programming W Editors W Table editors W Declaration editors
 Programming W Editors W General operation W Dialog boxes for supporting input
 Programming W Variables and data types W Variables W Variable remanence Real-time operating system W Method of operation W Module W data security W Power-off handling
 Real-time operating system W Method of operation W Module W data security W Power-on handling
 Programming W Libraries W IEC Check library

3.2 프로그램 코드에서 메모리 복사와 초기화

복사 메모리 장소

프로그래밍 언어와 사용 중인 환경에 의거하여, 복사 데이터를 사용할 수 있는 몇가지 방법들이 있다. Structured Text 프로그래밍 언어에서, 변수 A 의 데이터는 변수 B 에 복사 될 수 있다. 그러나 이 작업에서 소스와 타겟 데이터 타입이 완벽해야 한다; 그렇지 않으면, 컴파일러 에러가 발생한다.

하나의 데이터 타입의 데이터가 다른 데이터 타입으로 복사되어야 한다면, “brsmemcpy()” 평션을 사용하면 된다. 이 경우에는 소스 메모리와 타겟 메모리 주소 뿐만 아니라 복사되기 위한 byte 숫자까지 명시되어야 한다.


	선언	<pre>VAR aTarget : ARRAY[0..4] OF USINT; aSource : ARRAY[0..4] OF USINT; END_VAR</pre>
	프로그램 코드	<pre>brsmemcpy(ADR(aTarget), ADR(aSource), SIZEOF(aTarget));</pre>

표 28 brsmemcpy()를 사용한 메모리 부분 복사

복사되는 데이터 블록을 수용하기에 타겟 메모리가 충분히 있는지 확인해야 한다. sizeof()와 min() 평션을 사용하면 가장 작은 메모리 장소의 크기를 알 수 있다. 이렇게 하면 제한된 메모리만 타겟 메모리에 복사되도록 할 수 있다.


	선언	<pre>VAR aTarget : ARRAY[0..2] OF USINT; aSource : ARRAY[0..4] OF USINT; min_len : USINT := 0; END_VAR</pre>
	프로그램 코드	<pre>min_len := MIN(SIZEOF(aTarget), SIZEOF(aSource)); brsmemcpy(ADR(aTarget), ADR(aSource), min_len);</pre>

표 29 “min()”을 사용한 제한된 메모리 장소에 복사하기

메모리 장소 초기화하기

다른 구성과 데이터 타입을 가지는 메모리 장소는 변수가 선언될 때 초기화 된다. 프로그램에서 어떤 데이터 장소를 덮어 쓰는 것이 필요할 수도 있다. 프로그램 코드에서 brsmemset() 평션을 사용하면 된다.

	선언	<pre>VAR aTarget : ARRAY[0..4] OF USINT; END_VAR</pre>
	프로그램 코드	<pre>brsmemset(ADR(aTarget), 0 , SIZEOF(aTarget));</pre>

표 30 “brsmemset()”을 사용한 메모리 장소 초기화

**예제: 메모리 초기화하고 복사하기**

- 1) 두 개 변수 선언
두 변수 모두 사용자 정의 데이터 타입 “recipe_typ”으로 선언하라.
- 2) brsmemset() 평션을 사용해서 첫 번째 구조체 초기화
초기화는 반복실행되어서는 안된다. 초기 값은 0 으로 한다.
- 3) 데이터 복사
두 번째 구조체에 첫 번째 구조체 내용을 복사한다. brsmemcpy() 평션을 사용하라. 이 평션을 사용할 때 데이터의 길이에 주의해야 한다. 복사 과정은 커멘드 설정에서 한 번 수행된다.

4 라이브러리 사용하기

라이브러리는 소프트웨어를 작은 단위로 묶고 필요할때 언제든지 재사용할 수 있게 해준다. 하나의 라이브러리는 다수의 평선, 평선 블록, 상수, 데이터 타입들의 모음이다.

이번 장에서는 라이브러리의 기술 뿐만 아니라 어떻게 평선와 평선 블록을 정확하게 사용할 수 있는지에 대한 정보를 간단하게 설명한다. 그 다음 단계로, 사용자가 생성하거나 실행해왔던 다른 평선들을 저장할 수 있는 사용자 라이브러리를 생성하는 방법에 대해 살펴 볼 것이다. B&R 기초 라이브러리 예제 프로그램 사용법도 확인한다.



그림 21 라이브러리 사용하기

4.1 일반 정보

하나의 라이브러리는 평선, 평선 블록, 상수, 데이터 타입의 모음이다. Logical View 에 언제든지 라이브러리는 삽입될 수 있다. B&R 기초 라이브러리 또는 사용자로가 생성한 라이브러리들을 선택할 수 있다.



Project management W The workspace W Catalogs

평선

하나의 평선은 실제 평선 호출, 전달하는 파라미터, 리턴 값으로 구성된다. 평선이 호출되면, 파라미터들이 전달되고 리턴 값이 즉시 리턴된다. 다른 파라미터들은 평선이 호출된 다음 순서에 전달된다.



프로그램 코드

```
result := MIN(100,200);
```

표 31 두 파라미터가 있는 평선 호출

평선 블록

평선와 달리, 평선 블록은 하나의 값 이상을 리턴한다. 또한 “instance” 선언이 필요하다. 게다가, 평선 블록은 하나의 task 가 여러번 실행되며 작동한다.

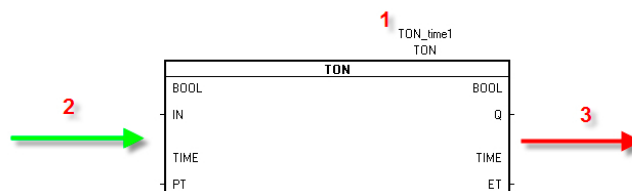


그림 22 (1) 인스턴스 변수, (2) 입력, (3) 평선 블록의 출력

- 1 인스턴스 구조체, 변수 선언 에디터에서 겹치지 않는 이름으로 선언되어야 한다.
- 2 입력 파라미터는 평선 블록이 호출되기 전이나 호출 되는 중에 직접 전달된다.
- 3 출력 파라미터는 평선 블록이 불러지는 도중 쓰이며 그 후에 프로그램 코드에서 사용될 수 있다.

다른 예시들은 다른 파라미터를 사용하여 task 안에서 계산이 가능하다.

인스턴스는 구조체의 한 부분으로 여겨진다. 평선 블록은 호출되는 순간 입력 파라미터를 얻고 그 후에 출력 파라미터를 인스턴스에 전달한다.





	선언	<pre>VAR TON_time1 : TON; TON_time2 : TON; END_VAR</pre>
	프로그램 코드	<pre>TON_time1(IN := enable1, PT := T#5s); TON_time2(IN := enable2, PT := T#10s);</pre>

표 32 다른 시간에 두 개의 TON 평선 블록 호출하기

두 타이머는 동시에 다른 지점에서 시작되고 다른 시간 주기 동안 동작한다. 이미 지나간 시간과 지연된 출력 신호는 평선 블록이 호출될 때 우선 저장된다.

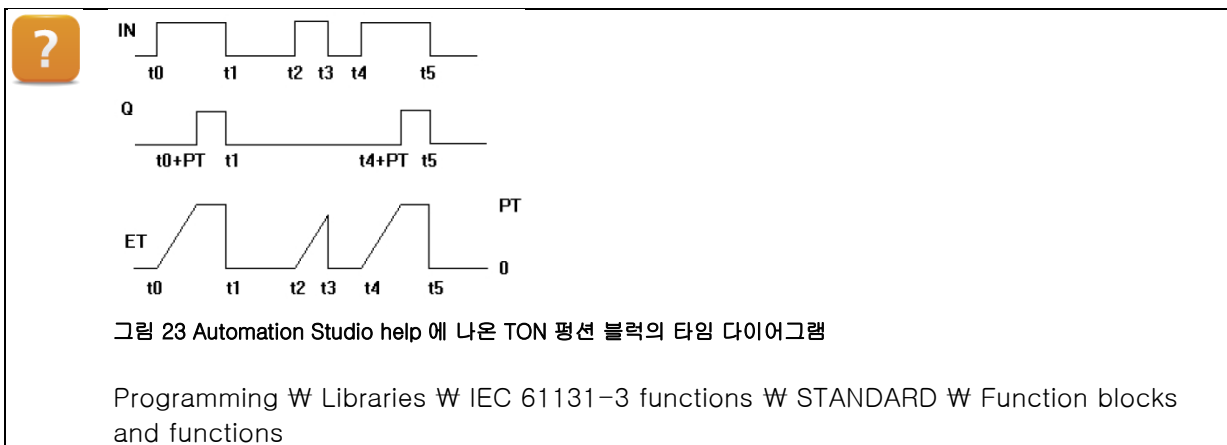
 평선 블록은 다음에 호출 될 때 오직 입력 파라미터만 얻는다. 평선 블록의 출력은 평선 블록이 호출될때 처리된다.

 평선와 평선 블록이 삽입될 때, “F1” 키를 누르면 Automation Studio 도움말을 열 수 있다. 도움말에서 파라미터 뿐만 아니라 특정 평선와 평선 블록의 입력과 출력에 대한 정보도 포함하고 있다.

예제: 클락 신호 발생시키기

“TON” 평선 블록을 호출하라. 1 초 클락 신호 그리고 1 초 정지 신호를 발생시켜라.

평선 블록 설명에서 타이밍 다이어그램을 보시오. 이 평선 블록이 출력 “Q”를 리셋시키기 위해서는 입력 “IN”에 적어도 한 번의 “FALSE” 신호가 들어와야 된다.



 Programming W Functions and function blocks W Functions

Programming W Functions and function blocks W Function blocks

평선 블록 호출하기: 입력 활성화 그리고 출력 상태


B&R 기초 라이브러리에 있는 모든 평선 블록들은 활성화 입력과 상태 출력이 있다.

활성화 입력은 평선 블록을 활성화 또는 비활성화로 만든다.

상태 출력은 현재 평선 블록의 상태를 나타낸다. 모든 평선 블록에 똑같은 상태가 몇 가지 있다. 다른 상태들은 다양하지만 평선 블록의 숫자 범위는 항상 평선블록들이 어떤 라이브러리에 속해있는지 명확하게 알아내는데 사용된다. 상태 값에 대한 정보는 Automation Studio 도움말에서 찾을 수 있다.


상수	에러 넘버	기능
ERR_OK	0	성공적인 실행, 에러 없음, 출력값 유효
ERR_FUB_ENABLE_FALSE	65534	활성화 설정 안됨, 평선 블록이 호출됐지만 실행 안됨
ERR_RUB_BUSY	65535	바쁨, 평선 블록이 호출됐지만 동작이 끝나지 않음, 다음 사이클에 다시 호출

표 33 평선 블록의 일반적인 상태 값




일반적인 상태 값들은 에러가 없고 프로그램 코드에서 리턴값이 평가될 때 정확하게 처리되어야 한다.

이런 세가지 상태는 “runtime” 라이브러리에서 자유롭게 정의된다. 이 라이브러리는 항상 Automation Studio 프로젝트에 포함된다.



Diagnostics and service W Error numbers W Libraries

Diagnostics and service W Error numbers W AR system W 65534 – 65535 Function block status



이번 예제는 상태 출력이 있는 평선 블록을 호출하는 방법을 보여준다. 정확하게 상태 출력을 평가하는 것은 필수이다. 호출된 평선 블록은 “AsARCFg” 라이브러리에 있는 “CfgGetIPAddr” 평선 블록이다.

선언	<pre>VAR GetIP : CfgGetIPAddr; sIPResult : STRING[20]; END_VAR</pre>
프로그램 코드	<pre>GetIP.enable := 1; GetIP.pDevice := ADR('IF3'); GetIP.pIPAddr := ADR(sIPResult); GetIP.Len := SIZEOF(sIPResult); GetIP();</pre>

```

IF GetIP.status <> ERR_FUB_BUSY THEN
  IF GetIP.status = 0 THEN
    (*everything ok*)
  ELSE
    (*place error handling here ... *)
  END_IF
END_IF

```

표 34 평선 블록 호출 후 상태 평가하기



상태가 BUSY 인 동안은 평선 블록 호출을 유지하는 것이 중요하다. 상태가 0 으로 변하자마자, 정확하게 실행되어온 절차와 출력 파라미터가 프로그램에서 사용된다. 0 또는 BUSY 이외의 다른 상태들은 프로그램 코드에서 정확하게 처리되어야 한다. 에러 넘버 개요와 관련된 내용은 라이브러리 문서에서 찾을 수 있다.

예제: 평선 블록 호출 및 상태 평가

1) 평선블록 “CfgGetIPAddr” 호출

평선블록 “CfgGetIPAddr”을 “AsARCfg”라이브러리에서 호출하라. 이 평선은 컨트롤러의 이더넷 인터페이스 IP 주소를 알아내는데 사용된다. 평선 블록은 관련된 인터페이스 이름이 필요하다. 이 이름은 Physical View 에서 관련된 인터페이스의 configuration 에서 볼 수 있다.

2) 상태 값 평가

정확하게 상태 출력을 평가하라. 변수 “status_signal”에 상태 = ERR_FUB_BUSY 일 때 1 의 값을 설정하라. 이 변수의 상태가 0 이면 2 의 값을 할당하라. 만약 다른 에러 (상태)가 발생한다면, 변수 값을 100 으로 설정하라.

3) 도움말에서 상태 값 검색

Automation Studio 도움말에서 이 평선 블록에 대한 에러 코드를 찾아라. 에러를 찾기 위해 도움말이 제공하는 제안을 살펴보자.

4) 이 에러는 어떻게 처리하는가?

이 에러를 프로그램이 어떻게 처리할수 있는가를 생각하라. 프로그램 코드 안에서 이에 대한 대책을 실행시킬 방법이 있는가?



Programming W Libraries W Configuration, system information, runtime control W As-ARCfg



Diagnostics and service W Error numbers W Libraries

라이브러리 컴포넌트

하나의 라이브러리는 몇 가지 컴포넌트와 특징으로 구성되어 있다.

다음의 컴포넌트들을 포함한다:

- .fun file: 인터페이스 또는 평선나 평선 블록 예시 구성을 포함한다.
- .var file: 평선 블록이 리턴할 수 있는 파라미터 뿐만 아니라 평선이나 평선 블록에서 기대되는 파라미터들의 상태를 포함하는 숫자 상수를 포함한다.
- .typ file: 평선 블록 내부에서 필요하거나 응용 프로그램에서 평선 블록에 전달되어야 하는 것들을 구조화한다.



그림 24 라이브러리 컴포넌트



Programming W Libraries

Project management W Logical View W Libraries

4.2 사용자 라이브러리 생성하기



프로그램 코드 재사용을 위하여, 첫 번째로 유지될 수 있는 자가 포함 모듈을 분할해야 한다. Automation Studio 에서 생성된 사용자 라이브러리는 이를 위한 이상적인 방법이다.

Phase 설계를 시작하기 전에, 평선과 평선 블록 같은 개별 기능 단위 관찰에 주의를 기울여야 한다. 이렇게 하면, 인터페이스가 선언될 수 있다. 마지막으로, 평선 범위를 구현할 수 있다.

그림 25 새로운 사용자 라이브러리

라이브러리를 생성시 다음 질문들에 주의해야 한다:

- 이 라이브러리의 기능은 무엇인가?
- 어떤 평선와 평선 블록이 필요한가?
- 평선 단위를 보기 위한 인터페이스를 어떻게 해야 하는가?
- 상수와 구조체가 사용될 것인가?
- 어떤 task 를 다루기 위해 다른 라이브러리에서 필요한 것이 있는가?
- 어떻게 라이브러리를 전달하거나 저장할 것인가?

사용자 라이브러리 삽입

새로운 라이브러리는 툴박스를 사용해서 추가하면 된다. 몇몇 카테고리들은 필터를 설정할 때 남겨진다. IEC 라이브러리를 설정하기 위해서, 결과 목록에서 알맞은 항목이 선택되어야 하고 Logical View 에 추가되어야 한다.

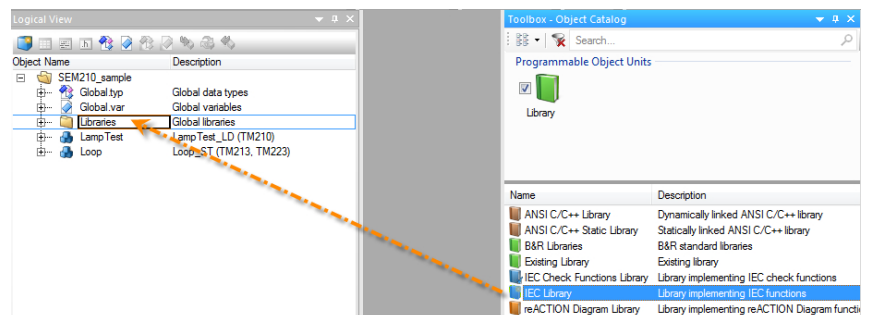


그림 26 툴박스를 통해 새로운 IEC 라이브러리 추가

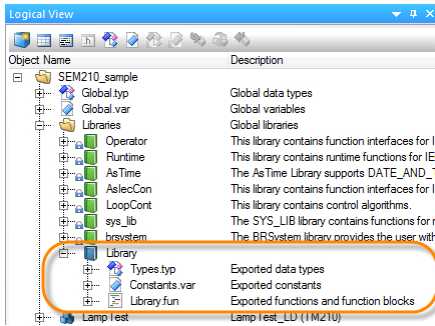


그림 27 새로운 IEC 라이브러리 컴포넌트

만약 새로운 IEC 라이브러리가 Logical View 에 추가되어진다면, 라이브러리의 이름을 바꿀 수도 있다.

새로운 라이브러리는 아래의 컴포넌트들로 구성된다:

- 라이브러리의 이름과 설명
- 상수 선언 (*.var)
- 데이터 타입 선언 파일 (.typ)
- 평선와 평선 블록 선언 파일 (.fun)

? Project management W Logical View W Libraries

평선 블록 추가하기

새로운 라이브러리가 생성된 후에, Logical View 에서 선택할 수 있다. 툴박스는 평선이나 평선 블록을 새로운 IEC 라이브러리에 추가할 때 사용된다. 이를 실행할 때, 평선 이름, 아이콘, 평선 블록 인터페이스를 명시하기 위한 wizard 가 나타난다.

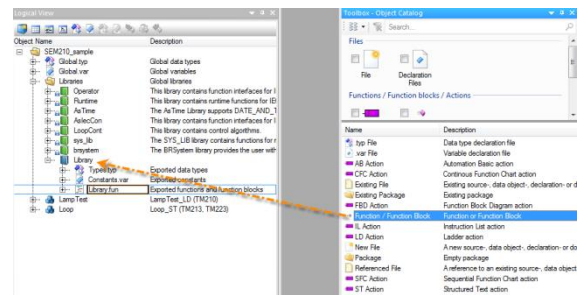


그림 28 새 라이브러리에 평선 블록 추가하기

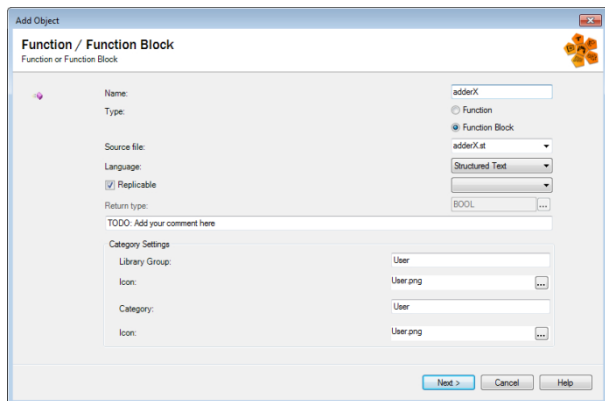


그림 29 카테고리에서 평선/평선 블록 선택

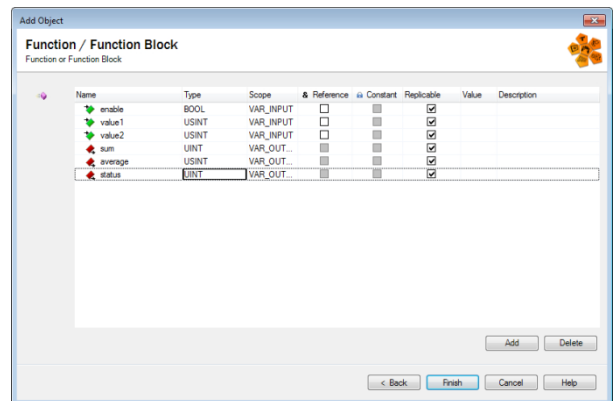


그림 30 평선 블록 인터페이스 구성

다음 설정들을 할 수 있다:

- 평선 또는 평선 블록의 이름과 설명
- 평선인지 평선 블록인지 선택
- 평선 또는 평선 블록의 프로그래밍 언어
- 기능: 리턴 값의 데이터 타입
- 입력과 출력 파라미터 선언

평선 또는 평선 블록이 추가된 후, *.fun 파일 내용에 따라 변경된다. 변경사항은 언제든지 *.fun 파일에 만들어진다.


게다가, 평선 또는 평선 블록의 이름을 가지는 소스 파일은 라이브러리에 삽입된다; 이 라이브러리는 실제 평선이 실행되는 곳이다.

이제 소스 파일에서 이 실행파일을 시행시킬 수 있다. 평선 블록 인터페이스의 파라미터는 변수로 사용할 수 있다.

평선 블록이 하나의 상태 값을 출력한다면, 이 출력 값에 대한 상수를 정의하는 것을 추천한다. 이 상수는 라이브러리의 .var 파일로 선언된다. 사용자 상수를 위한 정의된 사용자 장소가 있다.


Diagnostics and service W Error numbers W AR system W 50000 – 59999 user area

Automation Studio 도움말은 사용자 라이브러리 생성에 관한 종합적인 설명을 제공한다.


Programming W Libraries W Example for creating a user library

실행과 테스트

새로운 라이브러리 평선이 올바른지 확인하기 위해서, 반드시 테스트 되어야 한다. 이를 위해, 새로운 평선와 평선 블록은 어느 시스템에서든 호출되고 테스트 될 수 있다.


평선 블록 사용하기

다른 라이브러리로부터 평선 블록이 사용자 라이브러리에 사용되어야 한다면, 첫 번째로 인스턴스를 선언해야 한다. 외부 평선 블록에 대한 인스턴스는 사용자 평선 블록 인스턴스에서 외부 변수로 선언해야 한다.

Exporting 과 transferring

라이브러리가 완성되면, 버전 넘버가 할당된다, 이는 라이브러리 속성에 있다. 그 다음으로는 사용자 라이브러리로 추출(export)이 가능하다. 이는 Logical View 파일 메뉴에서 할 수 있다. 사용자는 소스 파일을 포함할 것인지 안할 것인지 선택할 수 있다.

만약 소스 파일이 포함되지 않는다면, 라이브러리를 더 이상 변경할 수 없다.


라이브러리는 프로그램처럼 Logical View 패키지 안에서 관리된다. 라이브러리는 사용되고 같은 패키지 안에서 프로그램처럼 저장된다.

예제: “myMath” 라이브러리 생성하기

“myMath” 사용자 라이브러리를 생성하라. 스케일에 무게를 더하거나 평균값을 계산하기 위해서 “adder”라는 평선 블록이 필요하다. 추가로, 최대 무게를 알 수 있어야 한다. 활성화 입력과 상태 출력은 평선 블록에 포함되어야 한다. 평선 블록 입력과 출력 목록은 아래 표에 있다.

이름	특성/값	데이터 타입
파라미터 (.fun)		
Enable	VAR_INPUT	BOOL
Weight1	VAR_INPUT	INT
Weight2	VAR_INPUT	INT
Weight3	VAR_INPUT	INT

Status	VAR_OUTPUT	UINT
Sum	VAR_OUTPUT	DINT
Average	VAR_OUTPUT	INT
Maximum	VAR_OUTPUT	INT

표 35 “adderx” 에서 인터페이스와 함수



Programming W Libraries W Example for creating a user library

Project management W Logical View W Libraries W Exporting a user library

Programming W Editors W Table editors W Declaration editors W Function and function block declarations

4.3 B&R 기본 라이브러리 예제

Task 를 실행할 때 B&R 기본 라이브러리는 종합적인 지원을 해준다. 라이브러리를 쉽게 사용 할 수 있도록, B&R은 몇 가지 라이브러리 예제를 제공한다. 기본 솔루션의 도움을 받아서 이 라이브러리에 있는 평션들을 더 효율적으로 사용할 수 있다.

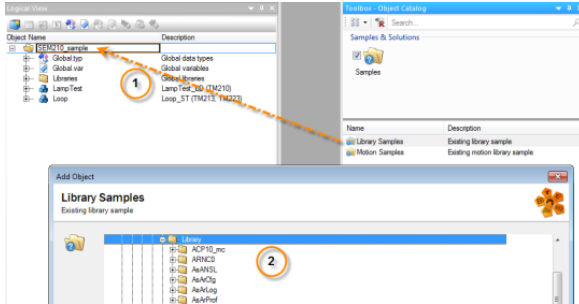


그림 31 톨 박스에서 라이브러리 예제 추가하기


이미 만들어진 예제는 Logical View 의 톨박스를 사용해서 import 할 수 있다.

- 1 예제 타입 선택
- 2 예제 설명 - 알맞는 라이브러리 분류

그 후 예제는 타겟 시스템에 직접 전달되기 전에 수정하고 명시된 응용프로그램(예: 인터페이스 파라미터 조정)에 알맞게 되어진다.

모든 예제는 시뮬레이션 모드에서 사용될 수 있다.

개별 프로그램의 구성은 같다. 이는 어떤 라이브러리가 다른 장소에 사용될 수 있는지에 대한 개요를 빠르게 제공한다. 예제에 대한 설명과 개요는 Automation Studio 도움말에서 찾을 수 있다. 예제 위치는 실제 라이브러리를 위한 참조문으로 같은 레이아웃을 가진다.

 Programming W Examples W Libraries
Programming W Examples W Adding examples

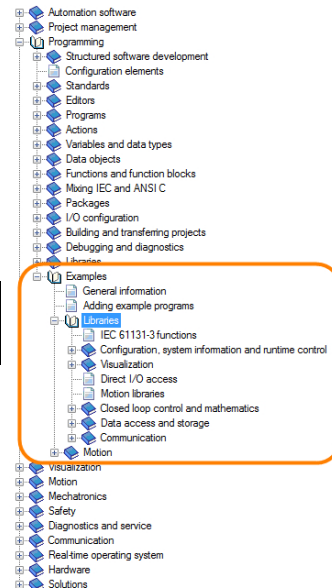



그림 32 Automation Studio
도움말에서 예제 관련 문서 개요

예제: 예제 삽입하기

아래 task 에서 우리는 Logical View 에 라이브러리 예제를 추가하고 실행할 것이다.

- 1) Logical View 에 다음 예제를 추가하라:

 Programming W Examples W Libraries W Configuration, system information, runtime control W Create and evaluate user logbook

- 2) 소프트웨어 configuration 에 할당
라이브러리 예제를 추가할 때, 사용자는 컨트롤러의 소프트웨어 구성에 관련된 프로그램이 적용되어야 하는지 결정해야한다. 이 메시지를 확인하면, 예제 프로그램은 디폴트 task 에 추가된다. 이 과제는 나중에 소프트웨어 구성에 드래그-앤-드롭으로 직접 변경할 수 있다.
- 3) 프로그램 전달
컨트롤러에 프로그램을 전달하라. 예제 프로그램에 따라, INIT 프로그램에서 몇 가지 파라미터를 조절하는 것이 필요하다. 예제 프로그램 설명에 이러한 파라미터에 대한 설명도 있다.
- 4) 프로그램 테스트
Automation Studio 도움말은 예제 프로그램을 작동시키기 위한 모든 파라미터와 커맨드에 대해 설명한다. 변수 모니터는 직접 커맨드를 설정해서 예제 프로그램을 동작시키는데 사용된다. 상태 값은 모니터링 과정 중 제공된다.

5 데이터 처리의 기초

지금까지 모든 변수, 배열, 구조체는 지역적으로 사용되었다. 그러나 메모리 한 부분에 있는 데이터를 파일로 자주 저장하거나 네트워크 상으로 전달하는 것이 필요하다. 이 경우, 몇 가지를 고려해야 한다.

데이터가 메모리에 어떻게 저장되는지 뿐만 아니라 어떤 형태로 저장되고 전달되는지 알아야 한다. 게다가, 데이터 내용은 저장되거나 전달되었던 구성 그대로를 유지해야 한다.

5.1 정렬

다른 처리 양식들은 데이터를 저장할 때 다른 규칙을 따른다. 갑작스런 예외 상황을 제외하고는 보통 사용자들이 이것들을 고려하지 않아도 된다. 그러나 다른 양식을 가지는 시스템에 데이터가 전달된다면, 그 데이터가 어떻게 저장되는지 알고 있는 것이 좋다.

예제: 사용자 정의 데이터 타입을 생성하고 메모리 사이즈를 알아보기

- 1) 사용자 정의 데이터 타입 수정
아래의 표처럼 "recipe_typ"의 기초 데이터 타입을 변경하라.
- 2) 메모리 사이즈 확인
사용자 데이터 타입 "recipe_typ"를 사용해서 변수 "SimpleCoffe"를 선언하라. Sizeof() 평션을 사용해서 이 구조체의 메모리 사이즈를 확인하라.
- 3) 결과 분석
당신이 기대했던 값과 일치하지 않는 데이터 길이가 나왔을 수도 있다.

요소 이름	데이터 타입
Price	USINT
Milk	UINT
Sugar	USINT
Coffee	UDINT

표 36 "recipe_typ" 데이터 타입의 요소

이전 task 에서 보여진 데이터 길이가 당신의 기대와 맞는가? 추가 데이터는 stuff 바이트 형태이다. 특정 프로세서 설계의 정렬을 위해서, 자동적으로 컴파일러가 추가한다. 프로세서가 더 쉽고 빠르게 저장된 데이터를 전달할 수 있도록 한다. 게다가, 홀수 길이의 메모리 주소로부터 짝수 길이의 데이터 길이를 가지는 데이터 타입을 프로세서가 읽어올 수 없다.

예제: 사용자 데이터 타입 수정하기

- 1) 요소 재배열
"recipe_typ"의 요소를 재배열하라. 이러면, 구조체 안의 데이터를 위한 stuff 바이트 일부를 사용할 수 있다.

5.2 데이터 포맷

데이터는 다양한 형식으로 저장되고 전달된다. 예를 들어 한 구조체를 보면, 그 구조체의 개별 바이트는 사용되고 있는 데이터 타입에 따라 해석되고 설명된다.

구조체의 데이터 내용이 한 바이트 필드에 복사된다면 2진 데이터만 남는다. 예를 들어서 사용자 데이터 타입 같은 조장 포맷이 알려진다면 데이터는 해석된다.

데이터가 파일에 저장되거나 네트워크로 전달된다면, 데이터가 저장되거나 보내는 곳뿐만 아니라 데이터를 궁극적으로 해석되어야 할 곳에서, 양쪽 모두 데이터 타입을 알아야 한다.

서로 다른 많은 데이터 형태들을 사용가능하다. 적절한 데이터 형태 없이 데이터는 2진 형태로 존재한다.



그림 33 데이터 형태를 알고 있는가?

	형식이 지정된 데이터	2진 데이터
2진	2B 34 33 37 37 34 38 36 35 38 36	2B 34 33 37 37 34 38 36 35 38 36
실수 데이터 타입	12.34	41 43 D7 0A
아스키(ASCII)	'Hello World!'	48 65 6C 6C 6F 20 57 6F 72 6C 64 21 00
XML	<?xml version="1.0"> <ComboBox> <Item ID="off"/> </ComboBox>	3C 3F 78 6D 6C 20 76 65 72 73 69 6F 6E 3D 22 31 2E 30 22 3E 0D 0A 3C 43 6F 6D 62 6F 42 6F 78 3E 0D 0A 3C 49 74 65 6D 20 49 44 3D 22 6F 66 66 22 2F 3E 0D 0A 3C 2F 43 6F 6D 62 6F 42 6F 78 3E 0D 0A

표 37 형식이 지정된 데이터와 2진 데이터



표에서 보는 것처럼, 모든 데이터는 이미 2진형태로 저장되어있다. 만약 데이터 형식이 알려진다면 다르게 표현될 수도 있다. 파일 내용에 대해서는 파일 확장자로 알 수 없다; 확장자는 데이터가 어떻게 해석되어야 하는지에 대한 간단한 정보만을 제공한다.

5.3 데이터 일관성

메모리의 내용 할당이나 “brsmemcpy()”를 사용해서 A 에서 B 로 복사된다면, 한 방향으로 하는 것이 더 빠르다. 같은 데이터가 파일에 저장되거나 네트워크로 전달어야 한다면, 데이터가 일관성을 유지하는 것이 중요하다.

데이터를 저장하거나 전달하는 것은 몇 가지 프로그램 사이클을 취하며, 한 사이클 당 얼마만큼의 양이 처리될 수 있는지 예측하는 것은 불가능하다. 이러한 이유로, 이 프로세스 동안 내용이 변하지 않는 것이 중요하다.

저장 또는 복사 절차 시작 전에, 데이터를 준비하는 프로그램에서 “preparation step”을 실행시키는 것은 좋은 생각이다. 이러한 준비 과정은 저장 또는 전달 과정이 실행되기 전 한번만 실행된다. 데이터 일관성은 이런 방법으로 확실히 될 수 있다.



저장되거나 전달될 때 데이터가 일관성을 유지하지 않는다면, 어떤 것도 들어있지 않는 데이터와 함께 저장되거나 전달될 수 있다. 예를 들어, 일련의 측정 값들이 더이상 연대순으로 맞지 않을 수도 있다. 프로세스에 따라서, 이것의 효과는 최소 또는 최대가 될 수 있다.



선언

```
VAR
    stepSave : UINT;
    aSend : USINT[0..9];
    aRaw: USINT[0..9];
    cmdSave : BOOL := FALSE;
```

END_VAR

프로그램 코드

```
CASE sSave OF
    0: (*--- wait for instruction*)
        IF cmdSave = TRUE THEN
            sSave := 1; (*--> Prepare & save*)
        END_IF
    1: (*--- prepare data*)
        brsmemcpy(ADR(aSend), ADR(aRaw), SIZEOF(aRaw));
    2: (*--- save data to file*)
        SaveData(ADR(aSend));
END_CASE
```

표 38 파일에 저장하기 전 데이터 준비하기

“cmdSave”를 호출하기 바로 전에, 수정된 “aRaw”데이터가 응용프로그램으로부터 바이트 필드 (“aSend”)에 복사된다. 이것은 데이터 구성을 확실시하는 전체 저장 과정²을 통해 똑같이 남는다.

² 이곳에서 호출된 평선은 상징적으로 시퀀스를 설명하는데 사용된다. 이는 실제로 존재하지 않는다.

6 데이터 저장하기과 관리하기

데이터는 수많은 방법으로 컨트롤러에 저장된다. 이는 DRAM 에서 메모리 블록처럼 관리되거나, 검사 함께 모니터를 사용한 B&R 데이터 오브젝트로 변환되거나 파일 시스템 또는 네트워크에 있는 파일에 로컬로 저장된다.

6.1 메모리 블록 예약하기

몇 가지 특이한 경우에서, 런타임에 DRAM 메모리 장소를 예약하는 것이 필요하다. 이 메모리는 사용자 프로그램에서 사용가능하다. 예약된 메모리 시작 주소를 사용해서 접속될 수 있다. 사용자에게 의해 다시 해방되지 않는 한, 이 예약된 메모리 장소는 시스템에서 사용할 수 없다.

예약된 메모리는 사용자 프로그램에서 사용할 수 있도록 시스템에서 연속으로 사용가능한 메모리 위치로 구성되어야 한다.



이 메모리의 내용은 사용자에게 의해 초기화 되어야 한다. 예약된 메모리 내용은 DRAM 에 있기 때문에, 컨트롤러가 재시작될 때마자 사라진다.

SYS_Lib 을 사용한 메모리 관리

예약을 위한 메모리 양이 컨트롤러가 부팅되기 전에 알려진다면, SYS_Lib 라이브러리에 있는 “tmp_alloc”과 “tmp_free” 평션을 이용할 수 있다.

이 평션들은 컨트롤러 프로그램 INIT 과 EXIT 서브루틴에서 호출된다.



Programming W Libraries W Configuration, system information, runtime control W SYS_Lib W Functions and function blocks W Memory management

AsMem 을 사용한 메모리 관리

AsMem 라이브러리는 시스템에서 메모리의 큰 부분을 예약하기 위해 프로그램 INIT 서브루틴에서 사용될 수 있다. 메모리 블록은 전환될 수 있고 사이클 프로그램에서 호출된 평션 블록 사용에서 해방될 수도 있다.



Programming W Libraries W Configuration, system information, runtime control W AsMem W Functions and function blocks
Programming W Examples W Libraries W Configuration, system information, runtime control W Managing memory areas

6.2 Technology Guard 에 데이터 저장

Reatin 변수와 영구 변수는 버퍼 메모리(배터리에 부착된 SRAM 또는 FRAM)에 저장될 수 있다. 이렇게 사용하여 중요한 변수 값은 시스템 재시작 후에도 사용할 수 있다. 그러나, 저장된 데이터는 항상 관련된 컨트롤러에 저장된다, 예를 들어, 런타임 유틸리티 센터 변수 서비스를 사용해서 백업과 재저장할 수 있다.

테크놀로지 가드(Technology Guard)는 컨트롤러에 데이터 저장을 위한 대안으로 사용된다. 라이선스 관리를 위해 작동시간 카운터와 200 바이트 데이터 저장의 두 가지 조작 보호를 제공한다. 이 기능들은 AsGuard 라이브러리의 평션 블록을 사용하면 접속할 수 있도록 만들어졌다. 테크놀로지 가드는 데이터가 포함된 다른 컨트롤러에 필요하다면 삽입할 수 있다.

테크놀로지 가드에서 다루지는 데이터 일관성이나 정렬과 관련된 데이터 관리는 사용자에게 책임이 있다.

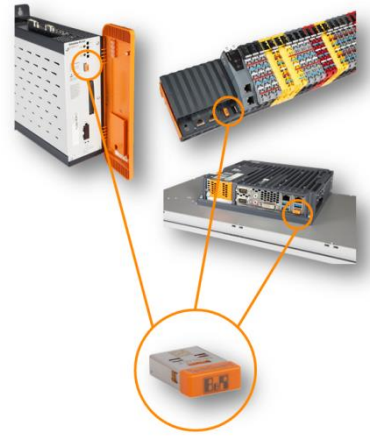


그림 34 데이터 저장매체, Technology Guard

?

Programming W Variables and data types W Variables W

- Variable remanence
- Permanent variables

Automation software W Technology Guarding

Programming W Libraries W Configuration, system information, runtime control W AsGuard

Programming W Examples W Configuration, system information, runtime control W Technology Guard examples

예제: 테크놀로지 가드에 데이터 저장

이번 예제의 목적은 시스템에 독립적인 방법을 사용해서 이전에 설정된 레시피 구성을 데이터로 저장하는 것이다. 테크놀로지 가드에 있는 메모리는 저장 위치로 사용된다. 예제에서는 AsGuard 라이브러리를 사용하여야 하며, 관련 라이브러리 예제를 참조한다. 예제를 위한 약간의 변화는 다음 목적을 위해 구성되어야 한다.

- 1) AsGuard 라이브러리에 대한 예제 추가
- 2) 컨트롤러에 예제 전송(transfer)
- 3) 삽입된 테크놀로지 가드가 읽어지는지 확인
- 4) 테크놀로지 가드에 저장하기 위해서 데이터 소스처럼 레시피 구성 전송
- 5) 테크놀로지 가드에 데이터 저장
- 6) 테크놀로지 가드에서 데이터 읽기
- 7) [추가 과제]

시간이 있다면, 다른 컨트롤러에 있는 응용 프로그램에 의해서 테크놀로지 가드의 데이터가 읽어지는지 확인하라.

6.3 데이터 오브젝트



소프트웨어의 다른 부분을 설정하기 위해서, 파라미터 파일의 파라미터 데이터를 사용할 수 있어야 한다. 필요하다면, 이런 파일들은 응용프로그램에서 읽거나 재저장될 수 있다.

B&R 데이터 오브젝트는 이런 경우에 사용할 수 있다. 파일을 저장할 수 있을 뿐만 아니라 Automation Studio 에서 직접 데이터를 쓸 수도 있다. 런타임 중에 데이터 오브젝트의 검사합계(checksum)을 볼 수 있다.

검사합계 모니터링은 손상된 데이터를 검출할 수 있도록 해준다. 이 정보는 log 에 저장된다.

그림 35 B&R 데이터 오브젝트

데이터 오브젝트는 DataObj 라이브러리 평션 블록을 사용하여 어플리케이션 프로그램에서 접근할 수 있다. 데이터 오브젝트는 이름을 확인하고, 읽고 쓰여진다. 데이터 오브젝트는 Logical View 에 추가된다. 예를 들어 파라미터의 기본 목록은 데이터 오브젝트에 저장된다. 데이터는 DataObj 라이브러리를 사용해서 구성을 읽고 복사할 수 있다. 런타임 동안 데이터 오브젝트에 생긴 변화는 Automation Studio 프로젝트에 적용될 수 없다.

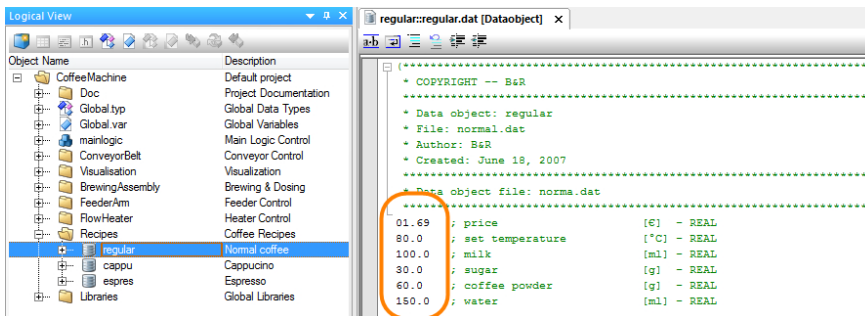


그림 36 기계 설정을 위한 기본 데이터 오브젝트

Programming W Data objects

Programming W Data objects W Simple data objects in B&R data object syntax


Programming W Libraries W Data access and data storage W DataObj

Programming W Examples W Libraries W Data access and data storage W Data storage


Real-time operating system W Method of operation W Module W data security

예제: DataObj 라이브러리 사용하기


- 1) Logical View 에서 데이터 오브젝트 생성
Automation Studio 의 Logical View 에서 데이터 오브젝트 “param”을 생성하라.
- 2) 구조체 “recipe_typ” 구조에서 데이터 입력
Automation Studio 도움말을 사용해서, 구조체 “recipe_typ” 형태에 데이터 오브젝트 데이터를 입력하라. 구조체 요소들의 데이터 타입, 사용한 데이터 타입, 정렬 등에 주의하라. 올바른 문법은 Automation Studio 의 도움말에서 찾을 수 있다.

 Programming W Data objects W Simple data objects in B&R data object syntax

- 3) 내용 읽어오기
평션 블록 “DatObjInfo()”와 “DatObjRead()”를 사용해서 데이터 오브젝트로부터 데이터를 읽어라. “DatObjInfo()” 평션 블록을 사용하여 데이터 오브젝트의 모듈 이름을 통해 데이터 오브젝트를 전달할 수 있다. 정보는 데이터 길이와 메모리 주소에 대한출력이다.
“DatObjRead()”를 사용하면 데이터 오브젝트로부터 데이터를 읽어올 수 있고, 메모리 주소에 복사할 수 있다. Logical View 에 DataObj 라이브러리 예제 프로그램을 import 할 수도 있다.

 Programming W Examples W Libraries W Data access and data storage W Data storage

- 4) 데이터 분석
불러온 데이터와 당신이 입력한 데이터 오브젝트의 데이터가 일치하는지 확인하라. 정렬에 대한 것에 유의하라!

 컨트롤러를 위한 프로그램처럼, 데이터 오브젝트는 컴파일러에 의해 컴파일되고 검사합계를 사용하는 2 진 오브젝트로 저장된다. 그러므로, ASCII 데이터와 달리 데이터 오브젝트에 저장된 데이터는 알아보기 쉽지 않은 형태로 되어있다. 만약 응용프로그램이 USB 플래쉬 드라이버나 네트워크 드라이버에 있는 파일 시스템을 사용하여 교환되기 위한 파일을 요구한다면 파일들은 데이터 오브젝트 대신에 사용될 수 있다. FileIO 라이브러리 평션 블록들은 파일로 작업할 때 사용된다.(“파일 시스템에 파일 저장”)

6.4 파일 시스템에 파일 저장

데이터가 항상 같은 시스템에 남아있는 것은 아니다. 파일은 동시에 데이터를 저장하거나 전달시키는 하나의 방법이다. FileIO 라이브러리는 컴팩트 플래쉬 카드, USB 플래쉬 드라이버 또는 네트워크 리소스에 있는 파일 시스템에 접속할 수 있도록 해준다.

결과는 다음과 같다:

- 컨트롤러의 파일 시스템에 접속
- 디렉토리 관리
- 디렉토리 검색
- 파일 생성, 읽어오기 그리고 쓰기
- USB 대용량 저장장치에 접속하기
- 네트워크에서 해방된 리소스에 접속하기
- 네트워크의 FTP 서버에 접속하기



그림 37 FileIO 라이브러리



파일 시스템에 접속하는 것은 설정된 파일 목적지 뒤에 상징적인 이름이 필요하다. 이것은 파일 디바이스로 참조된다. 파일 디바이스는 다음과 같은 이름이다(예: 특정한 위치에 있는 “F:\WRecipes\” “RECIPES”라는 디바이스 이름을 참조한다.).

파일 디바이스는 Automation Studio 시스템 구성 또는 “DevLink()” 평션 블록을 사용하여 런타임 동안 생성될 수 있다.

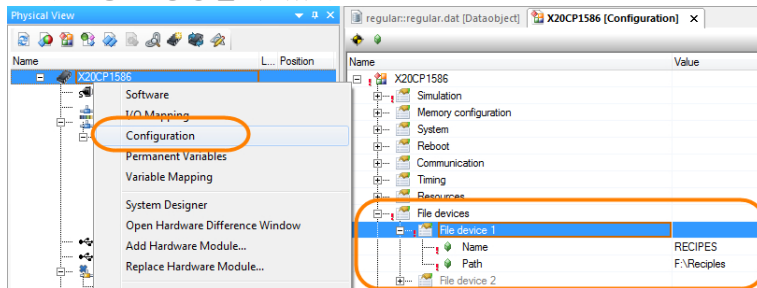


그림 38 시스템 구성에서 파일 디바이스 설정



Programming W Libraries W Data access and data storage W FileIO

Programming W Libraries W Data access and data storage W FileIO W General information W File devices, files and directories

Programming W Examples W Libraries W Data access and data storage W Data storage

예제: FileIO 라이브러리 사용하기

1) 텍스트 파일 생성


Notepad 와 어떤 타입을 가지는 텍스트 파일을 생성하라.


2) CompactFlash 카드에 텍스트 파일 복사


파일을 CompactFlash 카드에 복사하라. CompactFlash 는 런타임 동안 FTP³을 이용하여 접속할 수 있다. CompactFlash 카드는 카드 리더기⁴로 읽거나 카드 리더로 쓰여질 수 있다. Ar-Sim 또는 시뮬레이션 모드에서, “C” 드라이브는 사용자 컴퓨터 하드 드라이브⁵ 다.

3) 파일의 내용 읽기

상태 머신을 설정하라. 상태 머신의 스텝 넘버를 확인하는 열거형 데이터 타입을 정의하라. 파일을 여는 프로그램을 만들고, 내용을 읽고 다시 닫아라. 트레이닝 모듈 “TM230 – Structured Software Development”에는 머신 상태 프로그래밍에 도움을 줄 것이다. FileIO 라이브러리 예제는 Logical View 에 import 되고 활성화 할 수 있다.

 Programming W Examples W Libraries W Data access and data storage W Data storage

 CompactFlash 카드에 데이터를 저장할 때, 사용자 파티션에 저장하는 것이 중요하다. 일반적인 파일 시스템에서 “D:\W”드라이브이고, 안전한 파일 시스템에서 “F:\W”이다. 다른 파티션을 작동 중인 시스템이라면, 어플리케이션과 사용자 데이터는 명확하게 기술된다. CompactFlash 카드는 Automation Studio Runtime Utility Center 에서 만들 때 분할된다.

 Diagnostics and service W Service tool W Runtime Utility Center W Creating a list / data medium W Compact Flash functions

6.5 Mapp technology 레시피 관리

Mapp 테크놀로지⁶에서 우리는 사용자에게 통합 기능을 실행하는데 사용하기 쉬운 인터페이스를 제공한다. 레시피 데이터 로딩과 저장, 드라이브 축 제어와 프로세스 값 저장 등 복잡한 작업들은 사용하기 쉬운 mapp 테크놀로지 컴포넌트를 통해 실행된다.



그림 39 mapp technology logo

Mapp 테크놀로지는 설정과 프로그래밍을 통합한다. 기능은 기본 라이브러리를 사용하는 어플리케이션 프로그램에서 실행된다. 게다가 mapp 은 어플리케이션 소프트웨어에서 실행 설정없이 mapp 컴포넌트의 기능을 사용할 수 있는 설정 인터페이스를 제공한다.

³ “File Tranfer Protocol” FTP 규범. FTP 를 사용하여 접속하는 것은 컨트롤러의 Ip 주소를 요구한다. CompactFlash 카드에 있는 어떤 부분도 적절한 FTP 클라이언트 프로그램을 사용하면 접속할 수 있다.

⁴ Microsoft Windows 는 다수 파티션을 위한 제거가능한 드라이브에 “C”드라이브만의 접속을 허용한다. 그러나, Runtime Utility Center 는 다른 파티션으로부터 파일을 저장하고 재저장하는데 사용된다.

⁵ 시스템 파티션에 쓸 때 요구된 허락사항을 알게되는 것. 허용없이 데이터에 쓰기를 시도할 때, Microsoft Windows 는 임시 디렉토리에 데이터를 놓는다.

⁶ “Modular APPLication technology”에 대한 mapp 테크놀로지 규정



Application layer – mapp technology

- Concept
- Getting started
- Components

Mapp 테크놀로지를 사용한 레시피 관리

MpRecipe 컴포넌트는 빠른 속도로 간단한 레시피 관리를 위한 모든 기능을 제공한다. 이는 레시피 파일을 읽고 쓰는 것 뿐만 아니라 visualization 적용에도 연결된다.

MpRecipeRegPar 평션 블록은 레시피 관리를 위한 프로세스 변수 저장에 사용된다. MpRecipeXml 평션 블록은 파일 디바이스로부터 레시피 파일을 불러오거나 다시 저장하는데 사용된다. MpRecipeUI 평션 블록은 광범위한 기능을 위해 비주얼 컴포넌트에 있는 visualization 을 위한 완벽한 인터페이스를 제공한다.

개별 컴포넌트간의 커뮤니케이션은 MpLink 를 사용한다. 이는 Configuration View 에 레시피 컴포넌트를 위한 기본 구성을 추가하면 실행할 수 있다.

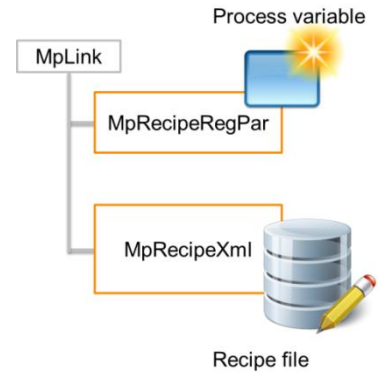


그림 40 레시피 관리를 위한 프로세스: 레시피 파일 저장과 불러오기

WebXs 를 이용한 진단

모든 Mapp 테크놀로지 컴포넌트는 웹 기반 진단을 제공한다. WebXs (Web Access)는 mapp 평션 블록의 입력 값, 출력 값, 상태 정보 등을 보여준다. WebXs(Web Access)는 System Diagnostics Manager 를 사용해서 관리할 수 있다.

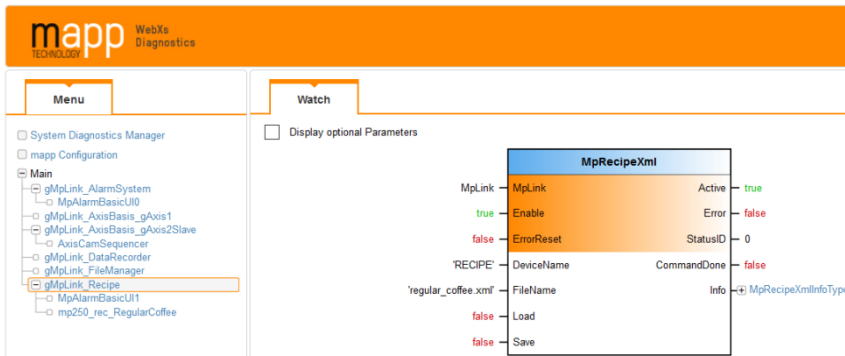


그림 41 자동 생성 WebXs 를 이용한 MpRecipeXml 컴포넌트 진단



Application layer – mapp technology

- Concept
- Getting started W Recipe management with visualization
- Components W Infrastructure W MpRecipe – Recipe management

예제: 레시피 파라미터로 레시피 구조 저장하기

레시피 파일을 저장하기 위해서, 레시피 컴포넌트에 있는 레시피 파라미터는 저장되어야 한다.

“RegularCoffee” 변수는 레시피 파일 데이터로 사용된다. “TM250 – Memory Management and Data Storage”에 있는 예제 “‘recipe_typ’ 구조체 생성하기” 안에 생성된 데이터 타입을 사용한다. 레시피 파라미터들을 성공적으로 저장한 후, 레시피 컴포넌트와 함께 레시피를 저장할 수 있다.

- 1) “recipe_typ”타입으로 “RegularCoffee” 변수 선언
- 2) 레시피 시스템 (MpRecipeRegPar)에 레시피 데이터 저장
- 3) 상태 ID 확인



1 어플리케이션 프로그램이 머신 상태 도움으로 생성된다면, 모든 mapp 평선 블록은 프로그램의 끝에 호출되도록 하라.
2 선택적 평선 블록 파라미터들은 확장된 기본 기능이 요구된다면 사용될 수 있다.

예제: 레시피 구조체 저장하고 불러오기

레시피 컴포넌트는 레시피를 저장하고 다운로드하는데 사용된다. 요구된 파일 이름과 확장자 그리고 존재하는 파일 디바이스를 명시하는 것이 필요하다. 이미 저장된 레시피 파라미터들은 똑같은 MpLink 변수를 사용해서 연결된다.

- 1) “CW:Recipe” 폴더에 “RECIPE” 파일 디바이스 설정
- 2) 레시피 컴포넌트(MpReipexml) 호출
- 3) 커맨드를 사용하여 레시피 컴포넌트 작동시키기
 - 레시피를 저장할 때 “cmdSaveRecipe” 사용
 - 레시피를 불러올 때 “cmdLoadRecipe” 사용
- 4) “CommandDone” 출력 값과 상태 ID 확인

예제: Enable 과 WebXs (Web Access) 확인

맵 컴포넌트는 자동 WebXs(Web Access)를 진단 목적을 위해서 제공한다. WebXs 는 소프트웨어 구성에 MpWebXs 가 할당되면 활성화 된다. 프로젝트가 전달될 때, 관련된 맵 컴포넌트 진단과 설정은 웹 브라우저에서 볼 수 있다.

- 1) Logical View 에 MpWebXs 라이브러리 추가
- 2) 소프트웨어 구성에 MpWebXs 할당
- 3) 프로젝트를 전달하고 컨트롤러 재시작
- 4) System Diagnostics Manager 를 사용해서 WebXs (Web Access) 호출
- 5) WebXs 에서 레시피 컴포넌트의 상태 ID 확인

6.6 데이터 베이스

데이터 오브젝트와 파일은 데이터를 쉽게 저장하는데 사용된다. IT 인프라구성에 직접 통합되기 위해서 databases 에 접촉하는게 필요하다. “AsDb” 라이브러리를 사용하면 SQL databases 에 연결할 수 있다.



Programming W Libraries W Data access and data storage W AsDb

7 데이터 전달과 커뮤니케이션

데이터 전달과 커뮤니케이션의 주제는 꽤 복잡하고 몇가지 다른 분야로 구성된다. Task 실행에 접근하기 전에, 관련된 다른 측면에 익숙해 지는 것도 좋은 방법이다.

데이터 전달과 커뮤니케이션에 관해 알아두어야 할 질문들:

- 어떤 데이터가 전달되어야 하는가?
- 어떤 스테이션과 커뮤니케이션 할 것인가?
- PC 또는 컨트롤러 인가?
- 어떤 매체가 사용될 것인가?
- 어떤 프로토콜이 사용될 것인가?
- 어떤 데이터 형식이 사용될 것인가?
- 다른 관련 플랫폼이 있는가?
- 요구된 프로토콜에 대한 기본 라이브러리가 있는가?
- 관련된 필드버스가 있는가?
- 얼마나 많은 데이터가 전달되는가?
- 전달 속도와 반응 시간과 관련되어 무엇을 살펴봐야 하는가?

이 질문 리스트들은 깊은 설명이 필요하다. 그래야 다음을 시작할 수 있다.

Automation Studio 도움말은 커뮤니케이션 사용의 여러 타입들에 대한 개요를 제공한다.

이는 전달 방법과 프로토콜 뿐만 아니라 구성과 함께 사용할 수 있는 라이브러리들에 대한 정보를 포함하고 있다.



?

Communication

Programming W Libraries W Communication

그림 42 Automation Studio
도움말 - Communication

7.1 커뮤니케이션에 관한 일반 정보

당신이 연결과 어떤 개별 컴포넌트들이 연결되어 있는지 자세히 살펴본다면, 다양한 토폴로지와 접속 방법 그리고 전달 프로토콜 등에 대해 알게 될 것이다. 토폴로지는 커뮤니케이션 노드에 사용되는 연결 타입을 나타낸다. 다양한 토폴로지들은 전달 매체에 따라 달라진다. 전달 프로토콜은 데이터를 전달한다.

접속 방법 (Layer 1 + 2)은 커뮤니케이션 네트워크에 있는 네트워크 스테이션의 좌표를 가리킨다. 네트워크에 있는 각 스테이션은 구별된 표식 (스테이션 또는 노드 넘버, IP 또는 MAC 주소)을 가진다. 매체는 주로 사용되는 토폴로지를 결정하지만, 접속 방법 또는 전달 프로토콜(Layer 3+4)이 필수적이진 않다.

OSI layer 모델⁷

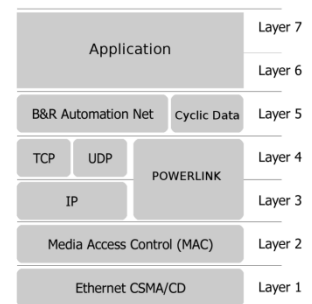


그림 43 TCP/IP 또는
파워링크 커뮤니케이션을
위한 OSI layer 모델

⁷ <http://en.wikipedia.org/wiki/OSI-model>

전달 매체

전달 매체는 커뮤니케이션 노드 사이의 물리적 연결 타입을 설명한다. 이는 케이블, 신호 레벨, 연결 (플러그) 등을 포함한다. 전달 거리는 다양할 수 있고 토폴로지는 매체에 영향을 받는다.

자주 사용되는 전달 매체:

- Serial(RS232 / RS485 / RS422)
- CAN
- Ethernet

Topologies (토폴로지)

토폴로지는 커뮤니케이션 노드간의 연결 타입이다. 가장 기본적인 형태로 커뮤니케이션 노드는 전달 매체에 직접 연결된다. 조금 더 복잡한 연결 타입은 조금 더 유연한 방법으로 커뮤니케이션 노드를 연결시킨다. 아래의 표는 일반적인 토폴로지와 지원되는 전달 매체 목록이다. 아래 그림은 개별 토폴로지의 기본 구성을 나타낸다.

토폴로지	지원되는 전달 매체
Point-to-point movement	Serial, CAN, Ethernet
Bus	CAN, RS485, RS422
Line	Ethernet
Tree	Ethernet
Star	Ethernet
Ring	Ethernet

표 39 토폴로지와 해당 전달 매체 개요

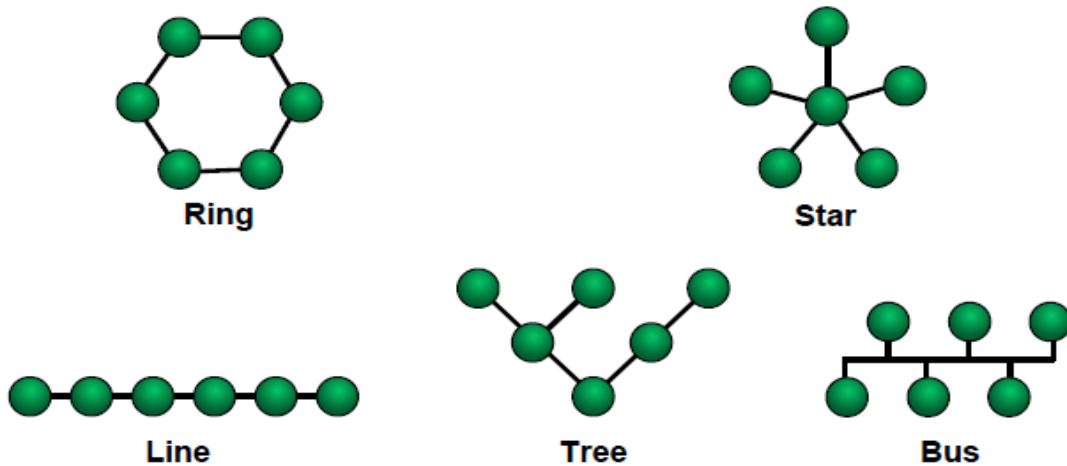


그림 44 주로 사용되는 토폴로지

접속 방법

하나의 네트워크에는 하나의 스테이션만 데이터를 보낼 수 있다. 이는 커뮤니케이션 매체에 어떤 좌표 스테이션 커뮤니케이션이 사용되어야만 하는 이유이다.

자주 사용되는 접속 방법:

- Master – Slave
차례차례 연결, 보통 master 와 slave 가 있다. Master 가 slave 에 커맨드를 보내고, 어떤 응답을 주고받거나 발생한다.

- Token Ring
Token Ring 네트워크, 하나의 token 사이클이 링을 통과한다. 이 token 은 다음 노드로부터 지나간다.
- CSMA CA / CD
네트워크는 일반적으로 몇가지 스테이션을 포함한다. 접속 방법에 따라, 네트워크 상의 스테이션은 다른 좌표를 가진다. 예를 들어, 충돌이 발생되어 발견되고 해결되는 접속 방법이 있다 (CSMA / CD)⁸. 또한 네트워크 충돌이 보호되는 방법이 있다 (CSMA / CA)⁹.
- Time slot method
Ethernet POWERLINK, Ethernet 전달 매체에 따르는 time slot 방식을 사용한다. 그러므로, 충돌을 발견하거나 방지할 필요가 없다. 이 전달 방법은 실시간이다.

필드버스 시스템

필드버스 시스템은 IO 시스템을 연결하거나 다른 장치를 컨트롤러에 연결할 때 사용한다. 필드버스 시스템의 정의는 주로 전달 매체, 지원되는 토폴로지, 접속 방법, 전달 프로토콜로 구성된다. 필드버스 장치는 Automation Studio 에서 삽입되고 설정된다. 이 장치의 설명은 보통 직접 automation Studio 에 import 될 수 있는 총괄적인 형태(.eds, .gsd, .xdd 파일)로 제공된다. 당신은 필드버스 통합 시스템에 대한 자세한 설명은 Automation Studio 도움말에서 찾아볼 수 있다.



Communication W Fieldbus systems

전달 프로토콜

전달 프로토콜은 데이터를 A 에서 B 로 전달하는 방법을 설명한다. 리시버는 전달 프로토콜이 받는 데이터를 정확하게 설명하기 위해 어떻게 구성되었는지 알아야 한다. 데이터는 올바른 형태로 전달되어야 한다.

전달 프로토콜 요소는 다음을 포함한다:

- 보내는 이의 정보
- 받는이의 도착 주소
- 데이터의 양
- 데이터
- 검사합계

전달 프로토콜 TCP/IP¹⁰ 은 네트워크에서 컨트롤러와 PC 사이에 커뮤니케이션을 위해 사용된다. CSMA/CD 접속방식을 사용하면 충돌을 발견하고 해결할 수 있다. 데이터는 빠르게 전달되지만, 실시간으로 작동하기 위해 충돌을 해결하기 위한 시간이 필요하다. Ethernet POWERLINK 는 컨트롤러, 원격 I/O 모듈, 드라이버, 통합 안전 테크놀로지를 연결하는데 사용된다. Ethernet POWERLINK¹¹ 에 의해 사용되는 Time slot 방법은 네트워크에서 발생하는 충돌을 막아준다. 그 결과로 데이터는 빠르고 실시간으로 전달된다.



Communication
Communication W POWERLINK

⁸ Carrier Sense Multiple Access / Collision Detection (CSMA / CD)

⁹ Carrier Sense Multiple Access / Collision Avoidance (CSMA / CA)

¹⁰ Internet Protocol (IP)에 의거한 Transmission Control Protocol (TCP)

¹¹ <http://www.ethernet-powerlink.org/>

7.2 커뮤니케이션 라이브러리

Automation Studio 는 커뮤니케이션에 관한 기본 라이브러리를 포함하고 있다. 모든 펄션 블록은 활성 입력과 상태 출력을 가지고 있다. 상태 출력 값은 상수로 미리 정의되고 관련된 라이브러리 도움말에서 설명을 볼 수 있다.

커뮤니케이션을 만들기 위해서는 첫 번째로 인터페이스를 열거나 초기화해야 한다. 전달을 위해서, TCP/IP 를 사용한 클라이언트 서버 커뮤니케이션처럼 다른 스테이션에 연결이 구성되어야 한다. 만약 커뮤니케이션이 더 이상 필요하지 않다면, 연결이 제거되고 인터페이스가 닫히며 관련된 시스템 리소스가 해방된다.

많은 어플리케이션 라이브러리 예제가 이용 가능하며 Automation Studio Logical View 에 추가 할 수 있다. 관련 설명은 Automation Studio 도움말에서 찾을 수 있다.

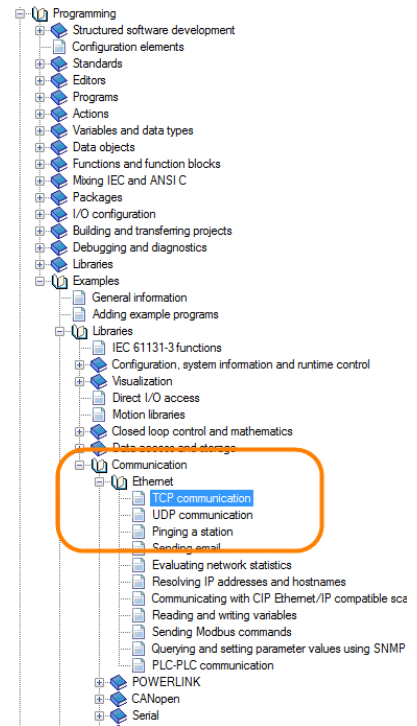



그림 45 Automation Studio
도움말에서 커뮤니케이션 라이브러리
예제

	Programming W Libraries W Communication
	Programming W Examples W Libraries W Communication
	Diagnostics and service W Error numbers W AR system

응용 가능성

커뮤니케이션 라이브러리 사용은 매우 많은 응용 시나리오를 가능하게 한다. 예를 들어, AsTCP 라이브러리는 TCP/IP 연결 설정과 모니터링을 할 수 있도록 해준다. 컨트롤러, IP 카메라, PC 같은 타사(3rd-party)장치를 B&R 컨트롤러에 연결할 수 있다.

아래 다이어그램은 커뮤니케이션 연결에 필요한 펄션 블록에 대한 기본 설명이다. 소켓은 서버 설정을 위하여 서버 스테이션에 연다. 또한 클라이언트는 소켓을 열고 서버에 연결한다. 두 스테이션의 커뮤니케이션은 send 와 receive 펄션을 사용한다. 이 펄션는 이미 “LibAsTCP1_ST” 라이브러리 예제에 구현되어 있으며, Automation Studio Logical View 에 추가할 수 있다.

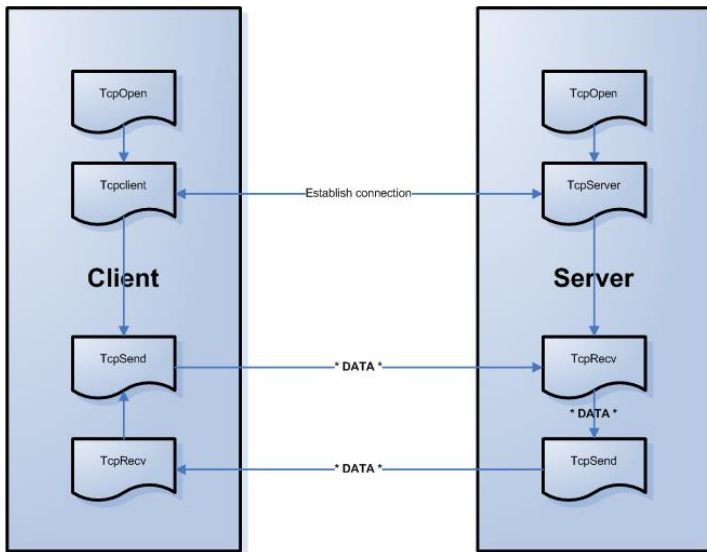


그림 46 AsTCP 라이브러리를 사용한 클라이언트-서버 커뮤니케이션과 연결을 보여주는 다이어그램

예제: TCP/IP 커뮤니케이션

- 1) 데이터 정의
컨트롤러 프로젝트로부터 데이터를 사용하라. 커뮤니케이션하기 위한 스테이션의 데이터와 구성과 데이터 형식이 같아야 한다.
- 2) Import 예제 프로젝트
Logical View 에서 “LibAsTCP1_ST”를 import 하라.
- 3) 당신의 커뮤니케이션 파트너에게 연결하라.
클라이언트와 서버프로그램이 로드될 스테이션을 선택하라. 그렇지 않으면 클라이언트와 서버 어플리케이션이 같은 컨트롤러나 시뮬레이션 환경이 로드 될 것이다.
- 4) 커뮤니케이션 테스트
클라이언트에서 서버로 그리고 그 반대로 정확하게 전달되고 있는지 확인하라. 당신은 변수 모니터나 변수 오실로스코프를 사용할 수 있다.
- 5) 커뮤니케이션 실패 시나리오에 대해 테스트하라.
각 컨트롤러들을 연결을 차례로 끊고 커뮤니케이션 프로그램에 의해 재연결 되는지 안되는지 확인하라.

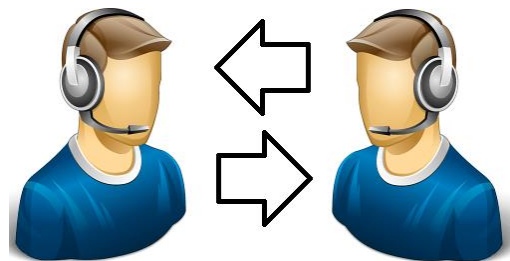


그림 47 TCP/IP 커뮤니케이션

8 요약

컨트롤 응용 프로그램 디자인에 책임이 있는 누군가는 끊임없이 데이터와 맞닥들인다. 데이터는 프로그래밍 언어와 관계 없이 변수와 상수로 표현된다. 다양한 기본 IEC 데이터 타입, 배열, 구조체 등은 이러한 목적으로 이용된다. Automation Studio 는 상수, 변수, 배열을 쉽게 초기화할 수 있는 사용자 편의를 위한 테이블 에디터가 특징이다. 평선 블록과 평선은 응용 프로그램에서 메모리 블록 관리를 위해 제공된다. 기본 데이터 처리 요소는 관련된 플랫폼에 데이터를 저장하는 형식이다. 평선은 프로그램에서 메모리 주소나 메모리 사이즈를 결정하고 사용할 수 있다.



그림 48 메모리와 변수



그림 49 데이터 처리와 저장



그림 50 커뮤니케이션과 데이터 변환

Automation Studio 는 데이터 오브젝트, 데이터, 프로세스 데이터 저장을 위한 데이터 베이스 접속을 제공한다. 다른 기계와의 통신은 지원하는 필드버스 시스템과 전반적인 B&R 컴포넌트의 네트워크 지원에 의해 가능하다. 광범위한 기본 라이브러리는 이러한 목적을 위해 제공된다. 도움말과 예제 프로그램, 컨트롤 응용 프로그램의 통합 시스템 기능 등은 이를 더 쉽게 도와 줄 것이다.