

논문 2005-42SC-3-4

PLC 출력 Unit의 출력 점수 확장에 관한 연구

(Study on Expansion of Output Point for PLC Output Unit)

유 정 봉*, 전 호 익**, 남 상 업***

(Jeong-Bong You, Ho-Ik Jeon, and Sang-Yep Nam)

요 약

PLC는 공장자동화 설계에서 가장 많이 사용되고 있으며, PLC 언어에 대한 연구도 활발히 이루어지고 있다. 본 논문에서는 PLC를 사용한 공장 자동화 설계에서 제어하고자 하는 출력 기기의 수가 증가되었을 때 출력을 확장하는 방법을 제안한다. 기본 베이스에 출력 유닛을 더 이상 장착할 수 없는 상태에서 출력 점수를 2~3점 확장하고자 한다면, 증설베이스와 파워서플라이, 증설케이블 및 출력 유닛을 별도로 장착하여야 한다. 그러면 비용이 상당히 추가되어야 한다. 본 논문에서는 출력을 소량으로 확장하는 방법을 제안한다. 즉, 출력 유닛의 점점을 입력받아 디코딩하는 확장모듈과 인코딩 및 디코딩하여 최종 출력점점으로 확장하는 프로그램 모듈을 설계하고, 실험을 통해 그의 타당성을 확인하였다.

Abstract

PLC is the most widely utilized among many sorts of existing controller for the design of process control system, and study about a PLC language is performed actively. In this paper, we proposed the method that we increase an output points when increased of the output machine which is going to control it in the FA design that used PLC. Output point is going to be extended with 2-3 points in the state that cannot equip output unit in basic base any more. Then an extension base, a power supply, an extension cable and an output unit must be equipped particularly. Then a cost must be added very much. This paper propose the method which extended an output in a small quantity. We designed the expansion unit that received the points of output unit and decoded the signal and the program module that finally extended output points by encoding and decoding, and we confirmed his feasibility through a experiment.

Keywords : PLC, Ladder diagram, Expansion module, Output unit, Decoder

I. 서 론

시스템이 대규모, 고도화, 복잡화에 따라 다양한 형태의 제어 시스템을 요구하고 있는 현대의 공정제어에서 PLC(Programmable Logic Controller)는 필수장비가 되었다.

초기 PLC가 개발된 이후로 마이크로프로세서의 발전에 의해 고도의 기능을 갖는 PLC가 출현하여 대규모의 입출력을 처리할 수 있고, 공정간의 유기적 연결을 위한 통신 네트워크에 의해 링크될 수 있으며, 다양한 산술 및 논리 연산기능을 가지게 되었다.

PLC는 공장자동화(Factory Automation, FA)의 요구에 맞춰 기능이 향상되었고, 적용 범위가 확대되었으며, 특히 프로세서 제어 분야에서 중요한 역할을 하게 되었다.^[1-4]

산업 제어 시스템에서 소프트웨어의 질적인 향상뿐만 아니라 개발 효율을 높이고 프로그램 기술을 향상시키고자 PLC 프로그램 언어의 표준이 나오게 되었다.

PLC는 IEC-1131-3의 국제 규격에 제시된 표준언어를 사용하게 된다. PLC의 표준언어는 IL (Instruction

* 정회원, 공주대학교 전기전자공학부
(Division of Electrical & Electronics Eng, Kongju National University)

** 정회원, 혜전대학 컴퓨터과
(Dept. of Computer Eng, Hyejeon College)

*** 정회원, 경문대학 정보통신과
(Dept. of Information & Communication Eng, Kyungmoon College)

접수일자: 2005년3월6일, 수정완료일: 2005년5월11일

List), ST (Structured Text)의 텍스트 기반언어와 LD (Ladder Diagram), FBD(Function Block Diagram), SFC(Sequential Function Chart)의 그래픽 기반 언어가 있다.^[5]

PLC의 하드웨어는 블록형 PLC와 모듈형 PLC로 나누어지게 된다.

블록형 PLC는 하나의 블록 안에 전원부, CPU부, 메모리부, 입출력부가 내장되어 있어 기능 및 입출력 점수를 확장할 때 증설 유닛을 추가해야 하기 때문에 확장성이 떨어진다.

모듈형 PLC는 기본 베이스에다 필요에 따라 여러 기능을 갖는 모듈을 부가하여 사용할 수 있고, 입출력 점수를 확장하기가 편리한 장점이 있어 중규모 이상의 제어기로 많이 사용하게 된다.^[6-7]

PLC를 설계하여 운전하다가 출력 기기의 수가 많아져 출력을 확장하고자 할 때는 기본 베이스에 출력 유닛을 추가할 수 있는 여분이 있거나, 출력 유닛에 출력 점수를 확장할 수 있는 여분이 있을 때는 출력점수를 확장하기가 용이하다. 그러나 출력 유닛에 출력점수를 확장할 수 있는 여분이 없을 때는 출력 유닛을 추가하여야 한다. 예를 들어 출력 유닛에 3점의 추가 여유분이 있을 때 5점의 출력기기를 추가하여야 한다고 하면 출력 유닛을 추가해야 한다. 또한 기본 베이스에 유닛을 추가할 여유가 없을 때 약간의 출력 점수를 추가해야 한다면 증설베이스와 전원부등을 추가해야 하기 때문에 비용이 상당히 증가하게 된다. 이러한 단점을 극복하기 위해 본 논문에서는 출력 유닛을 추가하지 않고 기본으로 설치되어 있는 출력 유닛에서 출력 점수를 확장할 수 있는 확장 유닛과 확장 프로그램 모듈을 제안하고, 그의 타당성을 확인한다.

II. 확장 유닛과 프로그램 모듈 설계

1. 일반적인 확장 방법

PLC를 설계하여 운전할 때 프로그램하고 디버깅시에 출력의 점수를 확장해야 하는 경우가 있다. 그러나 기본 베이스에 유닛을 추가할 여분이 없을 때는 일반적으로 증설 베이스와 출력 유닛을 추가하여 사용하고 프로그래밍 및 디버깅하게 된다. 이와 같은 방법은 출력의 점점번호도 추가한 출력 유닛의 번호로 부여되어 고정된 점점번호로 사용해야 하는 것이다. 또한 약간의 출력 점수를 추가하고자 할 때는 비용과 시간이 많이 들게 되어 비효율적인 방법이 된다.

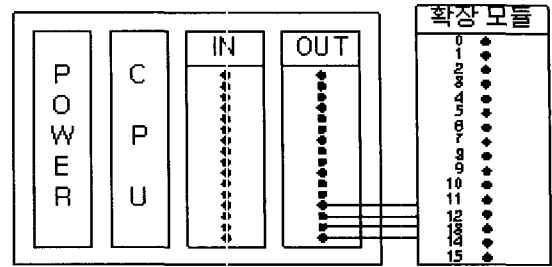


그림 1. 확장 시스템 구성도
Fig. 1. Configuration of System for Expansion.

2. 확장 유닛 설계

가. 확장 시스템 구성

일반적인 확장방법의 단점을 극복하고 설계의 효율성을 높이기 위해 본 논문에서는 확장 유닛과 프로그램 모듈을 제안한다.

확장을 위해 입출력 유닛을 장착한 시스템은 그림 1과 같다.

그림 1에서 기본베이스에 16점 출력 유닛을 장착하였을 때 상위 4비트(15~12번 점점)를 확장을 위한 점점으로 사용한다. 확장을 하지 않을 때는 일반 점점으로 사용할 수 있다.

확장을 위해 확장 유닛을 장착하게 되면 출력 유닛의 상위 4비트(15~12번 점점)를 확장 유닛에서 입력으로 받아들여 16개의 점점으로 확장시킨다. 따라서 그림 1의 예에서 출력 유닛을 0번 점점부터 26번까지 27개의 출력 점점을 사용할 수 있도록 확장 모듈 및 프로그램 모듈을 설계한다.

확장 모듈은 PLC 출력 유닛의 최상위 4비트의 출력 점점을 릴레이로 입력받는 PLC 인터페이스부, 4개의 점점을 16개로 디코딩하는 확장 디코더부, 각 점점 번호를 표시하는 7-세그먼트 표시부로 구성된다. 또한 프로그램 모듈에서는 이 확장 모듈로 입력되는 출력 유닛의 4개의 점점을 인코딩하고 이 인코딩된 점점을 다시 확장 모듈의 16개의 점점으로 사용할 수 있도록 디코딩하는 디코딩 부로 구성이 된다.

나. PLC 인터페이스 부

PLC의 출력 유닛에서 신호를 받아 디코딩 하기 위한 인터페이스부는 그림 2와 같이 PLC와 같은 종류의 릴레이 출력으로 설계하였다.

그림 2에서 확장 스위치(S1)는 확장 모듈을 사용하고 자 할 때 On하면 된다. 스위치를 Off 했을 때는 릴레이의 각 a 점점과 b 점점은 모두 동작되지 않도록 한 것

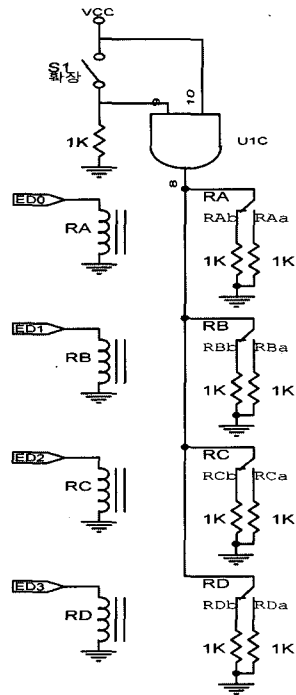


그림 2. PLC 인터페이스부
Fig. 2. Interface of PLC.

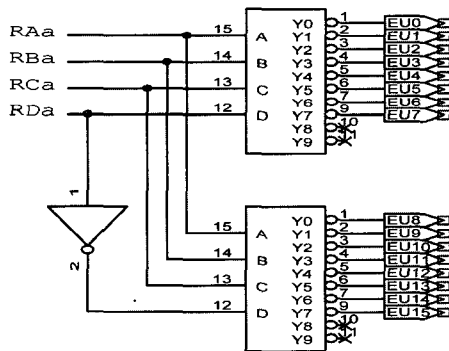


그림 3. 확장 디코더
Fig. 3. Decoder of Expansion.

이다. 이 것은 확장 모듈로써 동작시키고자 할 때만 사용하기 위한 것이다. 그리고 릴레이의 메인은 PLC에서 나오는 4개의 신호를 받아 16개로 디코딩하기 위한 것이며, 각 접점은 a 접점과 b 접점 모두 사용하고 있다.

다. 확장 디코더

PLC 출력 유닛의 4개의 접점을 받아 16개의 출력접점으로 확장하고자 한다. 디코더는 BCD 디코더 2개를 사용하여 상위 8접점, 하위 8접점의 16개 출력접점으로 확장한다. 확장 디코더의 회로는 그림 3과 같다.

라. 7-Segment 표시부

PLC에서 나오는 4개의 신호를 받아 16개로 디코딩하

표 1. 디코딩 값

Table 1. Value of Decoding.

10진수	PLC 신호	상위(S2)	하위(S1)
	DCBA	dcb	ab
0	0000	0000	0000
1	0001	0000	0001
2	0010	0000	0010
3	0011	0000	0011
4	0100	0000	0100
5	0101	0000	0101
6	0110	0000	0110
7	0111	0000	0111
8	1000	0000	1000
9	1001	0000	1001
10	1010	0001	0000
11	1011	0001	0001
12	1100	0001	0010
13	1101	0001	0011
14	1110	0001	0100
15	1111	0001	0101

기 때문에 최종 출력은 0번 접점부터 15번 접점까지 구성된다. 따라서 각 접점의 번호를 7-세그먼트로 표시하기 위한 것으로 두 자리로 표시된다.

(1) 상위(S2) 7-세그먼트

이중 상위 7-세그먼트는 0일 때는 공란으로 하고, 1의 숫자만 키면 되기 때문에 표 1과 같이 10에서 15까지의 숫자일 때만 1로 표시한다.

표 1에서 10진 숫자 0에서 9까지는 상위(S2)의 세그먼트는 공란으로 하며, 10진 숫자 10에서 15까지는 1로 표시한다. 따라서 상위 세그먼트 디코더인 74LS47의 a(1번) 신호에 다음의 식으로 표시되는 신호를 인가해야 한다.

$$a = BD + DC \tag{1}$$

(2) 하위(S1) 7-세그먼트

하위 세그먼트는 0에서 9까지만 표시하면 되고 10진 숫자 10에서 15까지는 다시 0에서 5까지만 표시하면 된다. 따라서 이 것을 논리식으로 표현하면 식(2)와 같다.

$$\begin{aligned} a &= A & b &= \overline{DB} + DC\overline{B} \\ c &= \overline{DC} + CB & d &= DC\overline{B} \end{aligned} \tag{2}$$

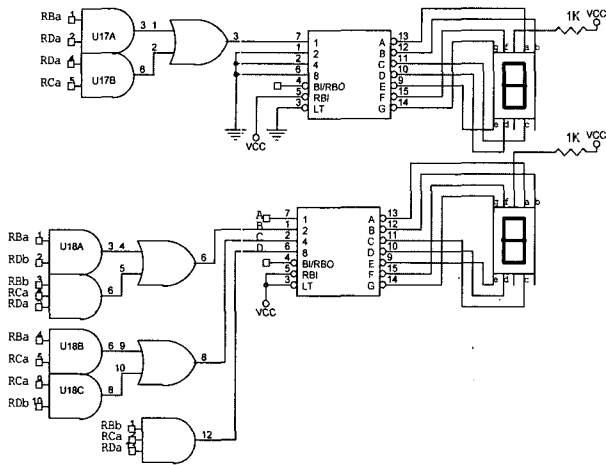


그림 4. 7- 세그먼트 표시부
Fig. 4. Display of 7-Segment.

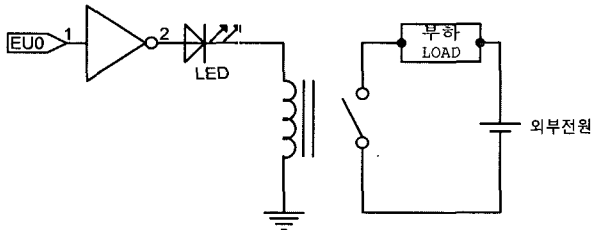


그림 5. 확장 출력접점
Fig. 5. Output Point of Expansion Unit.

이와 같이 식 1과 식 2에 의해 표시부를 설계하면 그림 4와 같다.

마. 확장 출력 접점

그림 3의 디코더 출력을 받아 외부기에 연결해 주기 위한 확장 출력접점으로 릴레이 출력으로 설계하였다. 확장 출력접점은 그림 5 같다.

그림 5는 EU0 하나만 나타내고 있으며, EU1에서 EU3까지의 회로도 동일한 형태로 된다.

3. 프로그램 모듈 설계

기본베이스에 16점 입력 모듈 1개와 출력 모듈 1개로 구성된 시스템을 예로 하여 프로그램 모듈을 설계하였다. 프로그램 모듈의 순서도는 그림 6과 같다.

16점 출력 모듈에서 확장하고자한다면 0~11번 접점은 그대로 사용하고, 12~15번 접점을 확장모듈의 입력으로 사용하면 된다. 프로그램 시 확장모듈의 변수는 ESW12~ESW27까지로 16개를 사용한다. 그러면 16점 출력모듈을 0~27번까지 28점 출력 모듈로 확장할 수 있게 된다.

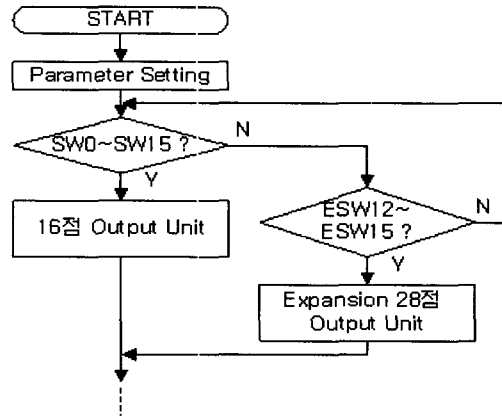


그림 6. 프로그램 모듈 순서도
Fig. 6. Flow chart of Program Module.

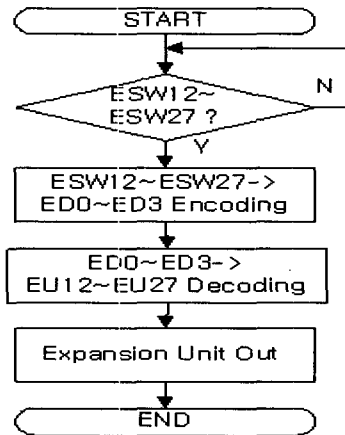


그림 7. 확장모듈 순서도
Fig. 7. Flow chart of Expansion Module.

그림 7에서 ESW12~ESW27을 ED0~ED3으로 인코딩하고, 확장모듈의 입력으로 입력하여 16개의 디코딩 출력으로 출력시킨다. 인코딩 된 ED0~ED3의 식은 (3)과 같다.

$$\begin{aligned}
 ED0 &= ESW13 + ESW15 + ESW17 + ESW19 \\
 &\quad + ESW21 + ESW23 + ESW25 + ESW27 \\
 ED1 &= ESW14 + ESW15 + ESW18 + ESW19 \\
 &\quad + ESW22 + ESW23 + ESW26 + ESW27 \\
 ED2 &= ESW16 + ESW17 + ESW18 + ESW19 \\
 &\quad + ESW24 + ESW25 + ESW26 + ESW27 \\
 ED3 &= ESW20 + ESW21 + ESW22 + ESW23 \\
 &\quad + ESW24 + ESW25 + ESW26 + ESW27
 \end{aligned}
 \tag{3}$$

또한, ED0~ED3을 에디터 프로그램에서 확인할 수 있도록 EU12~EU27로 디코딩한 식은 (4)와 같다.

$$\begin{aligned}
 EU12 &= \overline{ED3} \cdot \overline{ED2} \cdot \overline{ED1} \cdot \overline{ED0} \\
 EU13 &= \overline{ED3} \cdot \overline{ED2} \cdot \overline{ED1} \cdot ED0 \\
 EU14 &= \overline{ED3} \cdot \overline{ED2} \cdot ED1 \cdot \overline{ED0} \\
 EU15 &= \overline{ED3} \cdot \overline{ED2} \cdot ED1 \cdot ED0 \\
 EU16 &= \overline{ED3} \cdot ED2 \cdot \overline{ED1} \cdot \overline{ED0} \\
 EU17 &= \overline{ED3} \cdot ED2 \cdot \overline{ED1} \cdot ED0 \\
 EU18 &= \overline{ED3} \cdot ED2 \cdot ED1 \cdot \overline{ED0} \\
 EU19 &= \overline{ED3} \cdot ED2 \cdot ED1 \cdot ED0 \\
 EU20 &= ED3 \cdot \overline{ED2} \cdot \overline{ED1} \cdot \overline{ED0} \\
 EU21 &= ED3 \cdot \overline{ED2} \cdot \overline{ED1} \cdot ED0 \\
 EU22 &= ED3 \cdot \overline{ED2} \cdot ED2 \cdot \overline{ED0} \\
 EU23 &= ED3 \cdot \overline{ED2} \cdot ED1 \cdot ED0 \\
 EU24 &= ED3 \cdot ED2 \cdot \overline{ED1} \cdot \overline{ED0} \\
 EU25 &= ED3 \cdot ED2 \cdot \overline{ED1} \cdot ED0 \\
 EU26 &= ED3 \cdot ED2 \cdot ED1 \cdot \overline{ED0} \\
 EU27 &= ED3 \cdot ED2 \cdot ED1 \cdot ED0
 \end{aligned}
 \tag{4}$$

본 프로그램 모듈에서 ED0에서 ED3의 변수를 설계하고자 하는 PLC에 맞게 바꾸어 주기만 하면 어떤 PLC에도 적용이 가능하다.

III. 실험 및 결과분석

1. 실험

제작된 실험용 PLC 셋트는 그림 8과 같으며, 확장모듈은 그림 9와 같다.

PLC는 LG 산전의 GLOFA GM4 기종의 CPU를 사용하여 시뮬레이션하고 연구의 타당성을 확인하였다.^[8-9]

SW0~SW11까지는 입력 유닛의 입력 접점을 받아 출력 유닛의 0번~11번(%QX0.1.0 ~ %QX0.1.11)까지 출력시킨다. 이것은 그림 10과 같다.

확장 모듈 프로그램은 서브루틴으로 사용하였으며, 인코딩 입력 ED0~ED3은 그림 11과 같다.

또한 확장 모듈의 최종 디코딩 출력 프로그램은 그림 12와 같다.

그림 12의 EU12~EU27은 확장 모듈의 최종 출력 접

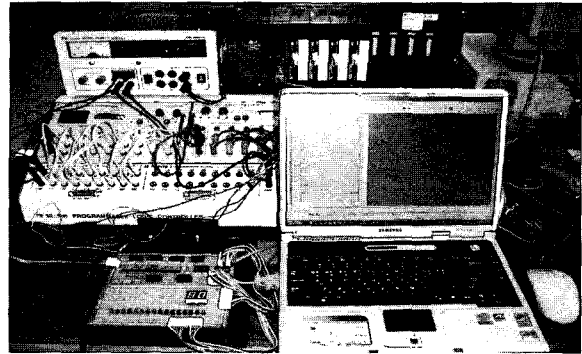


그림 8. 실험용 PLC 셋트
Fig. 8. PLC for Experiment.

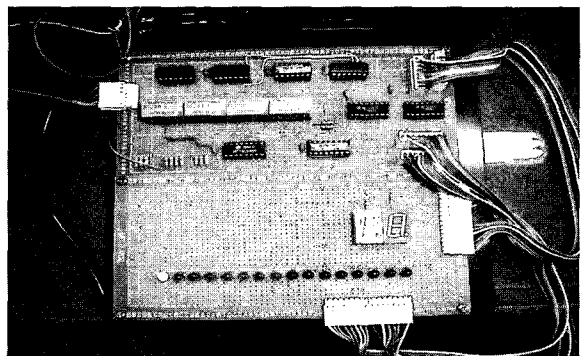


그림 9. 확장 모듈
Fig. 9. Expansion Module.

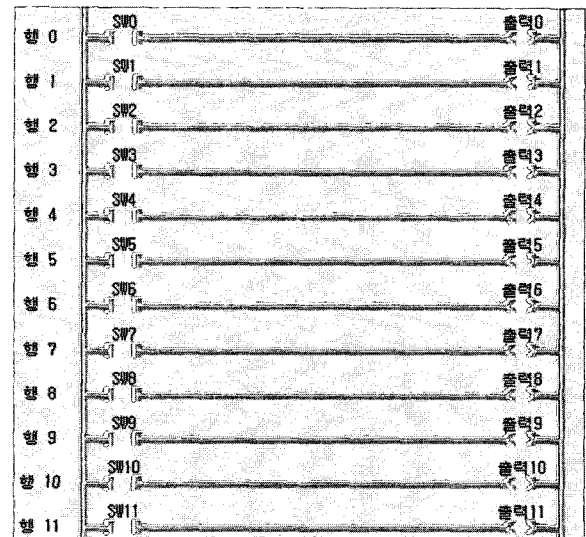
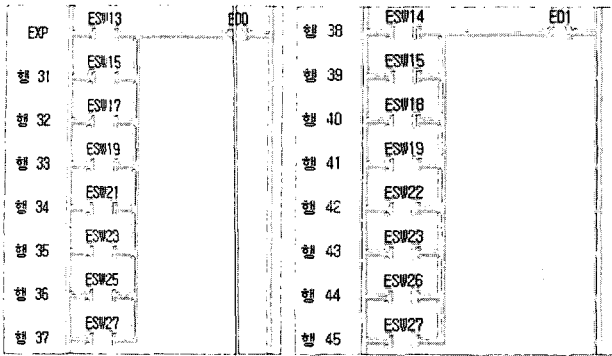


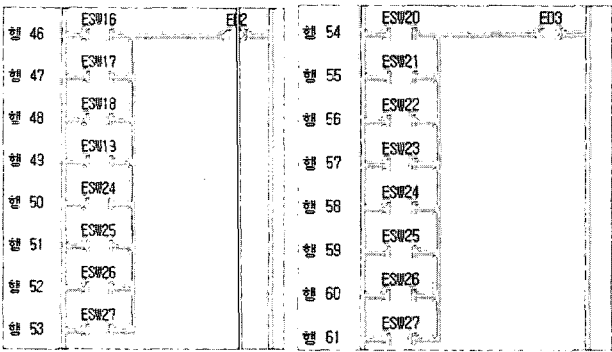
그림 10. 0번 ~ 11번 입출력 접점
Fig. 10. Input/Output Point of No0~No11.

점을 나타낸다. ESW12 접점을 On하였을 때는 확장 모듈의 EU12번 출력이 On이 되어야 한다. 이는 사진 6에서 알 수 있다. 그림 13의 (a)는 모니터링 화면을 나타내고 (b)는 확장모듈의 사진을 나타낸다. (b)에서 7-세그먼트는 '00'을 나타내고 0번 LED가 On 되어 있음을



(a)ED0

(b)ED1

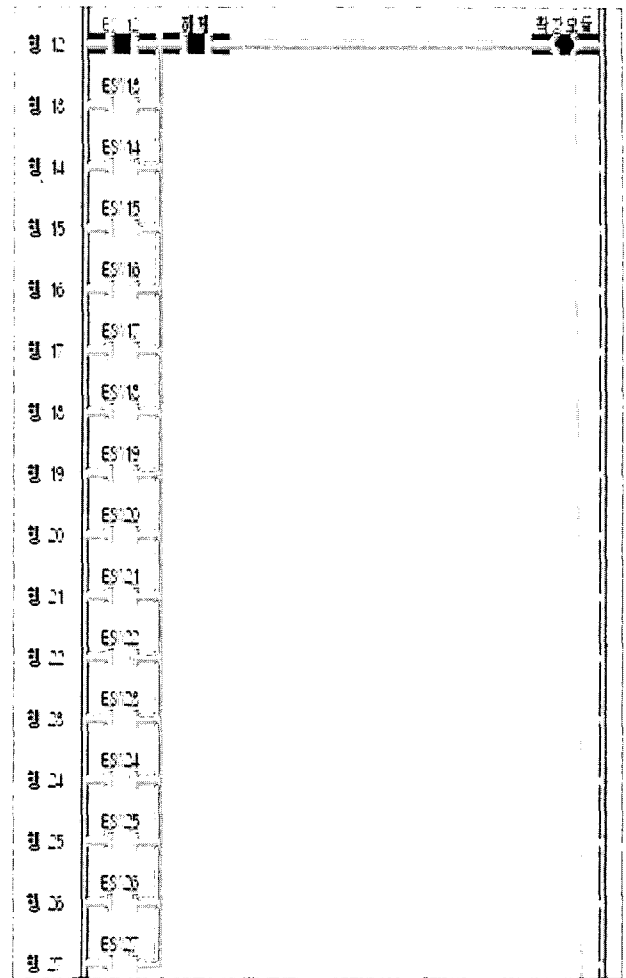


(c)ED2

(d)ED3

그림 11. Encoding 입력

Fig. 11. Encoding Input.



(a)모니터링

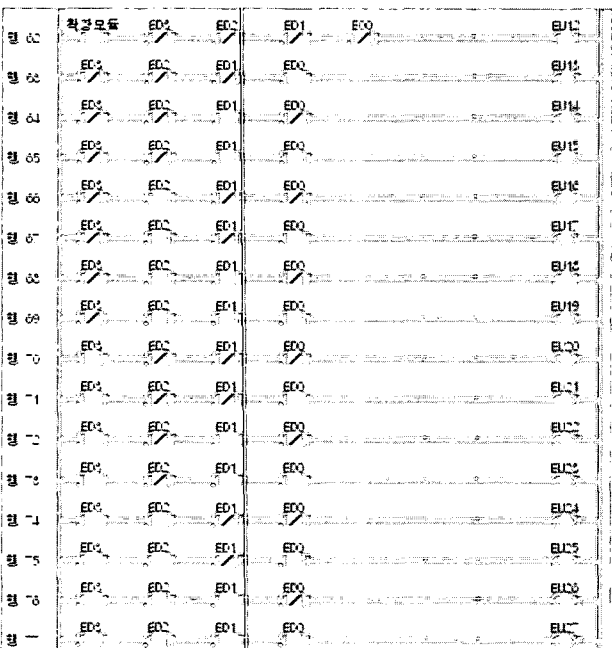
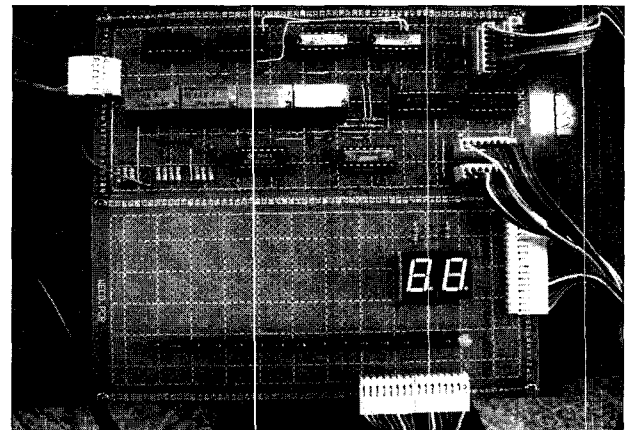


그림 12. 디코딩 출력 프로그램

Fig. 12. Program of Decoding Output.



(b)확장 모듈

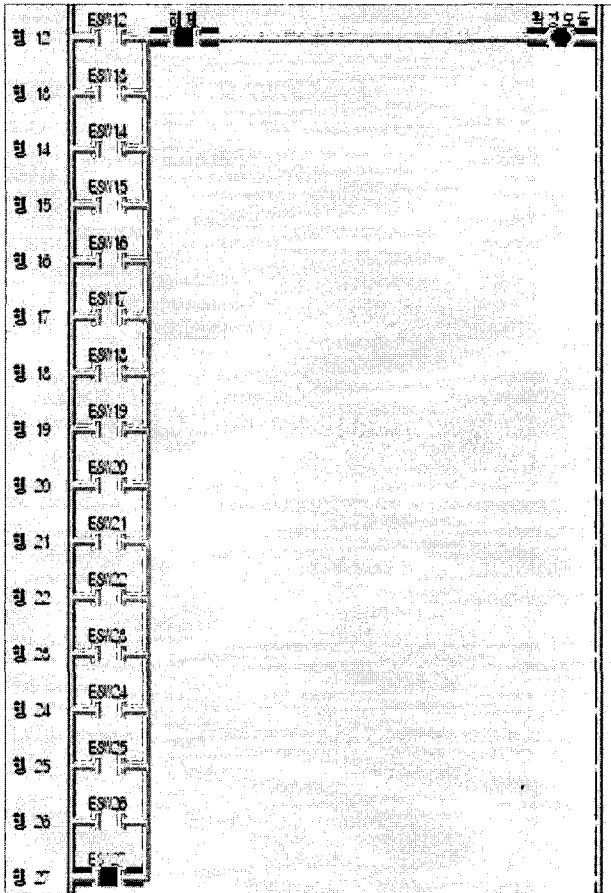
그림 13. ESW 12가 On 일 때

Fig. 13. When ESW 12 is turned On .

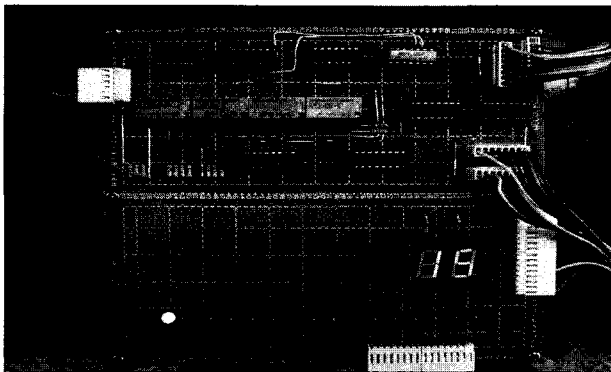
알 수 있다.

또한 그림 14에서는 ESW 27이 On일 때 모니터링 화면과 확장모듈의 사진을 보여주고 있다. (a)의 모니터

링 화면에서 ESW27 점점이 On 일 때 (b)의 확장 모듈에서는 7-세그먼트는 '15'를 표시하며, 15번 LED가 On 되어 있음을 알 수 있다.



(a)모니터링



(b)확장 모듈

그림 14. ESW 27이 On 일 때
Fig. 14. When ESW 27 is turned On.

2. 결과분석

2절 가항의 확장 시스템 구성에서 설명된 출력 유닛 확장 모듈의 하드웨어를 본문에 따라 설계하였고, 출력 유닛 확장 모듈의 입력과 출력의 접점을 제어할 수 있는 프로그램 모듈을 설계하였다.

프로그램 모듈에서는 PLC에서 처리된 ESW 12~ ESW 27을 ED0~ED3으로 인코딩하여 확장 모듈로 입

력 시킨 후 확장 모듈에서 EU12~EU27까지 디코딩하게 된다. 이에 대한 실험의 결과가 그림 10에서 그림 14까지 보여준다.

일반적인 확장방법을 사용하면 접점번호는 그대로 0번 ~15번을 사용해야 한다. 그러나 본 논문에서 제안된 방법을 사용하게 되면, 0번~27번까지 접점 번호를 확장할 수 있는 장점이 있다. 이 출력 유닛 확장 모듈과 프로그램 모듈은 일반적인 확장방법에 비해 출력접점을 폭넓게 사용할 수 있는 장점이 있다.

IV. 결 론

본 논문에서 PLC 출력 유닛의 출력 접점을 확장 모듈에서 입력 받아 프로그램 모듈에서 인코딩 신호로 변환하고 다시 디코딩 신호로 변환한 후 확장 모듈로 출력 점수를 확장할 수 있는 방법을 제안하였다.

PLC를 설계하여 운전하다가 소량의 출력 점수를 확장하고자 할 때 본 논문에서 제시한 방법을 사용하면 비용절감 및 시간절약측면에서 상당한 효과가 있을 것으로 판단된다.

본 논문에서는 LG GLOFA 기종의 PLC를 사용하여 실험을 하였지만, 프로그램 모듈에서 사용되는 변수의 값을 적용하고자 하는 PLC에 맞는 변수의 값으로 변경하여 주면 모든 PLC의 출력 유닛에 연결하여 사용할 수 있다는 큰 장점이 있다.

본 논문에서 제시한 방법은 PLC를 사용한 자동화 설계에서 프로그래밍 시 및 디버깅 시에 매우 효율적으로 사용할 수 있다.

참 고 문 헌

- [1] IAN G. Warnock, "Programmable Controllers Operation and Application", Prentice Hall, 1992.
- [2] Masaharu Oku, Takao Kokubu, Shigeru Masuda, and Kenzo Kamiyama, "Application of the Encapsulated Actuator Model to the Sequential Control Machines", Short Papers IEEE/ASME Transactions on Mechatronics, Vol 1, No 4, pp290-294, Dec 1996.
- [3] T. Kouthon, M. A. Peraldiabd, J. D. Deco-tignie, "Distributing PLC Control", International Conf. On IEC, IEEE 21'st Vol 2, pp. 1614-1619, 1995.
- [4] M. Zhou and E Twiss, "Design of Industrial automated systems via relay ladder logic programming and Petrinets", IEEE Trans on

- Systems, Man and Cybernetics-part C: Applications and Reviews, Vol 28, No 1, pp 137- 150, 1998.
- [5] R. W. Lewis, "Programming Industrial Control Systems Using IEC1131-3", The Institution of Electrical Engineers, 1992.
- [6] IAN G. Warnock, "Programmable Controllers Operation and Application", Prantice Hall, 1992.
- [7] 지유양, 유수복, "글로벌 PLC 프로그래밍 실습" 기전연구소, 2002.
- [8] "Mitsubishi PLC Programming Manual", Mitsubishi, QnA series, 2004.
- [9] "LG Programmable Logic Controller Glofa-GM" LG Industrial Systems, 2004.

 저 자 소 개



유 정 봉(정회원)

1988년 단국대학교 전자공학과 학사.

1990년 단국대학교 전자공학과 석사.

1998년 단국대학교 전자공학과 박사.

1990년~1993년 (주)신도리코

1999년~현 재 공주대학교 전기전자공학부 조교수

<주관심분야 : PLC 제어, BLDC 모터제어, 공장 자동화 알고리즘 설계>



남 상 엽(정회원)

1982년 단국대학교 전자공학과 학사.

1984년 단국대학교 전자공학과 석사.

2003년 단국대학교 전자공학과 박사.

1984년~1992년 삼성종합기술원 정보시스템 연구소

1992년~1998년 모토로라 반도체 통신(주)

1998년~현 재 경문대학 부교수

<주관심분야 : 멀티미디어 신호처리, 영상/음성인식, 임베디드시스템>



전 호 익(정회원)

1984년 단국대학교 전자공학과 학사.

1986년 단국대학교 전자공학과 석사.

1998년 단국대학교 전자공학과 박사.

1992년~현 재 혜전대학 컴퓨터과 부교수

<주관심분야 : 공장자동화 알고리즘 설계>