'87 韓國自動制御學術會議論文集 1987. 10.16~17

마이크로파스칼에 의한 디지탈제어시스템의 체계적 구성

우 광 방 김 현 기 °김 복 기
연세대학교 전기공학과

The Systematic Organization of Digital Control System by Micropascal

Kwang Bang Woo Hyun Ki Kim Bohk Ki Kim
(Department of Electrical Eng. , Yonsei University)

ABSTRACT

This study presents the methodology for the hierachical construction of digital control system which can be of practical value to a wide application of sequence control. Optimization of digital control system, programming of the system by micropascal, and simulation of a model program under test are discussed.

1. 서 론

본 논문은 광범위한 순차제어분야에서 응용할수 있는 디지탈제어시스템의 체계적인 구성 방법을 제시한다. 종래의 단순한 제어대상이 복잡하고 다양해짐에 따르는 문제점을 디지탈시스템의 체계화와 계층화로 해결하는 것이다. 전체적인 구성은 첫째 디지탈제어시스템의 제어형식 최소화 작업과 둘째 최소화된 제어형식을 마이크로파스칼로 프로그램하여 컴파일하고 마지막으로 프로그램을 모의 실험장치에서 시뮬레이션하는 것으로 되어 있다.

디지탈시스템의 제어상태 최소화 작업은 1) 순차제어를 체계적 상태 머신(Algorithmic state machine) 에 적용하여 2) 디지탈 제어시스템을 논리함수로 표현하고, 3) 논리함수를 이진프로그램으로 실현하고 , 4) 이진프로그램을 P-함수기법으로 최적그래프를 구성한다.

마이크로파스칼 프로그램과 컴파일은 1) 작성된 최적그래프를 마이크로파스칼로 프로그램하고 , 2) 마이크로파스칼 프로그램을 컴파일하여 이진 코드를 얻는다.

모의 실험은 이진 코드의 프로그램을 모의 실험장치에 로딩하여 시뮬레이션한다.

2. 체계적 상태머신과 P-함수 이론

체계적 상태 머신(Algorithmic state machine)은 제어장치가 궁극적으로 입력으로부터 받은 신호를 제어하여 출력을 내는 과정으로 생각하여 그림 1 과 같이 연산장치(Operation)와 제어장치(Control)의 두 부시스템으로 상호협력하여 제어되는 장치이다.

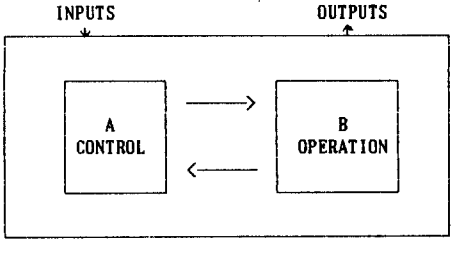


그림 1. Glushkov가 제안한 모델
Fig. 1. The Glushkov model

다음 P-함수는 논리함수의 하드웨어적 실현과 소프트웨어적 묘사를 최적화하기위한 수학적 방법이다.

정의 1 P-함수 : 스위칭 함수의 쌍<g,h>가 식 1 을 만족시키면 f의 P-함수 라고 하고 <g,h>▽f로 표기한다.

$$f * g = h * g \qquad \text{식 1}$$

정의 2 P°-함수 와 Prime P°-함수 : 차수 0의 P-함수은 h=0 또는 h=1인 P-함수<g,h>이다. Prime P°-함수은 다른 P°-함수의 정의역 함수 g^{*}에 포함되지 않는 정의역 함수 g를 가진 P°-함수이다.

정리 1 합성 법칙 T_i° :

$\langle g_0, h_0\rangle \triangledown f$ 와 $\langle g_1, h_1\rangle \triangledown f$ 에서

$\langle g_0, h_0\rangle T_i^{\circ} \langle g_1, h_1\rangle = \langle g_0 x_i' + g_1 x_i, h_0 x_i' + h_1 x_i\rangle \triangledown f$

정의 3 P^{β}-함수 과 Prime P^{β}-함수 :

$\langle g_0, h_0\rangle$과 $\langle g_1, h_1\rangle$를 각각 차수 p_0, p_1 ($p_0 p_1 < q-1$ 이고 $p_0$or $p_1 = q-1$) 인 P-함수이라고 하면 P^{β}-함수은 $P^{\beta\top}$-함수(r>1)의 정의역 함수에 포함되지 않는 정의역 함수를 가진 합성 법칙 T_i°에 의한 P°-함수이다. Prime P^{β}-함수은 다른 P^{β}-함수의 정의역 함수 $g^{\prime}$ 에 포함되지 않는 정의역 함수 g 를 가진 P^{β}-함수 $\langle g , h\rangle$이다.

3. 마이크로파스칼

마이크로파스칼은 그림 2 에서 구문그래프에 의해 설명될 수 있는데 여기에서 직사각형(예, "statemant")은 비말단 기호이고 타원(예, "endprogram")은 말단 기호이다.

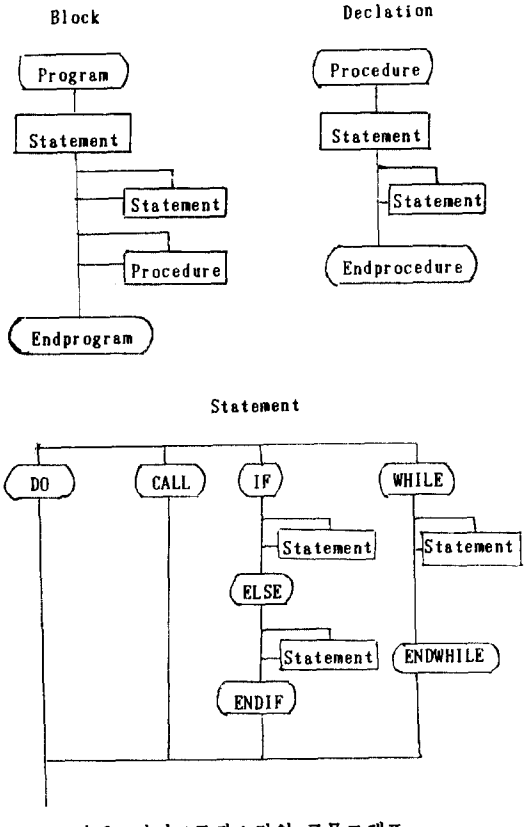


그림 2. 마이크로파스칼의 구문그래프
Fig. 2. Syntax graph for Micropascal

마이크로파스칼은 조합과 순차 논리를 묘사하기에 충분하며, 그리고 프로그래머를 콘트롤러의 프로그램에 적합하고 ,구조적 프로그램을 실현할 수 있다.

11 개의 주요단어는 표 1과 같이 주어진다.

표 1. 마이크로파스칼의 11개 명령어
Table 1. Eleven-line set of commands for Micropascal

Mnemonics	의 미
PROGRAM np	프로그램의 시작
END PROGRAM	프로그램의 종결
PROCEDURE p	부프로그램의 시작
END PROCEDURE	부프로그램의 종결
DO REG(j) < OUT	수행 명령
CALL p	부프로그램 호출 명령
IF x	입력 변수 조건 판정 명령
ELSE	부입력의 가지 명령
ENDIF	조건 판정명령의 종결
WHILE x	조건 반복 명령
END WHILE	조건 반복 명령의 종결

4. 모의 실험장치와 컴파일 알고리즘

모의 실험장치는 마이크로파스칼 프로그램을 컴파일하여 생성한 기계어코드를 입력으로 하여 결과의 테스트 작업을 용이하게 하는 고급언어 프로그래머블 콘트롤러이다.

이 모의 실험장치는 기존의 다른 머신들과 달리 하드웨어로 구성되어진 것이 아니고 구성의 흐름을 AHPL 언어를 사용하여 표현한 후에 C-언어로프로그램한 것이다 .

모의 실험장치에서 사용되는 언어는 하위레벨 언어로써 4개의 명령어들로 구성되며 다음과 같이 정의한다.

1) 출력 명령 (DO REG(j) <- OUT) : j 는 레지스터 REG의 번지이고, OUT는 출력상태를 나타낸다

0001	j3:0	OUT3:0

2) 호출 명령 (CALL ADSP) : ADSP는 부프로그램 SP의 처음 번지이다.

1100	___	ADSP7:0

3) 분기 명령 (IF x_iELSE ADR0) : i는 분기변수의 첨자이고 ADR0는 x_i = 0 일때 분기될 번지이다.

0000	i3:0	ADR07:0

4) 복귀 명령 (RET) : 부프로그램 SP에서 주프로그램 PP로 돌아오게 하는 명령이다.

1010	___	_____

그림 3 은 모의 실험장치의 개략도를 나타낸다.
이 모의 실험장치를 AHPL 로 표현된 것이 그림 4에 있다.

컴파일 과정은 고급언어인 원시프로그램(source program)을 동등한 하위레벨 언어인 목적프로그램(object program)으로 변환하는 것이다.

표 2 는 고급언어와 하위레벨 언어프로그램을 대응시켜서 11 개의 주요 단어를 4 개부분으로 나눈 것이다.

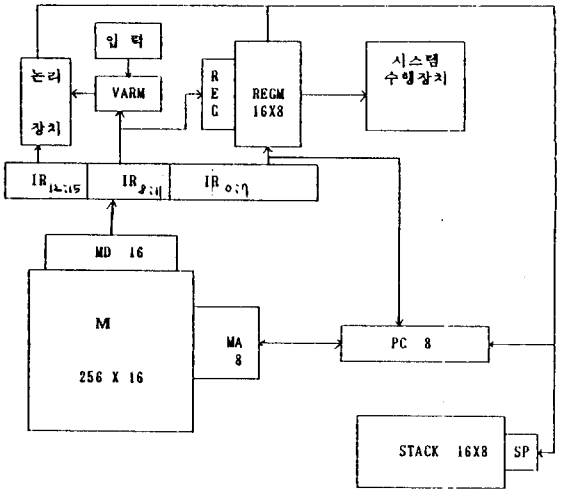


그림 3. 전체 모의 실험장치의 개략도
Fig. 3. The general organization of modelling test unit

```
MODULE :   Modelling test unit

MEMORY : AC[1];MD[16];PC[8];IR[16];STACK[16,8];
         M[256,16];  REG[4];VAR[4];  REGM[16,8];
         VARM[16,1];SP[4]
       INPUT  : start
       BUSES  : ABUS[16];BBUS[16]

1.  —> (SYN(start)) / (1)
2.   MA <— PC
3.   MD <— BUSFN ( M;DCD(MA))
4.   IR <— MD
5.  —> (IR15) / (15)
6.   NO DELAY
    —> (IR13) / (12)
7.   VAR <— IR5:4
8.   AC <— BUSFN ( VARM;DCD(VAR))
9.  —> (AC) / (11)
10.   PC <— INC(PC);
    —> (2)
11.   MA <— IR7:0;
      —> (3)
12.   REG <— IR9:8
13.   REGM*DCD(REG) <— IR7:0
14.   PC <— 8 T 0      ;
      —> (2)
15.  —> (IR14) / (18)
16.   SP <— INC(SP)
17.   PC <— BUSFN(STACK;DCD(SP))
     —> (2)
18.  —> (IR13) / (22)
19.   STACK*DCD(SP) <— PC
20.   SP <— DEC (SP)
21.   PC <— IR7:0
      —> (2)
22.   DEAD END
```

그림 4. 모의 실험 장치의 AHPL 프로그램
Fig. 4. The modelling test unit AHPL program

표 3은 IF.. ELSE.. ENDIF 의 알고리즘이고 표 4는 WHILE.. ENDWHILE 의 알고리즘이다.

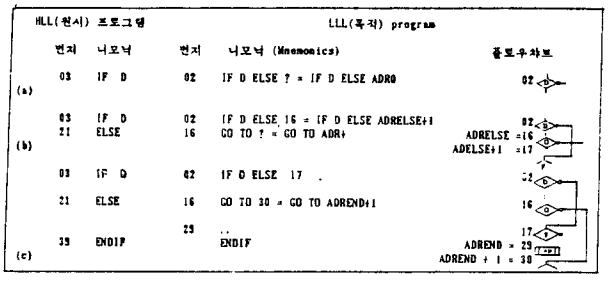
표 2. 마이크로파스칼의 4 가지 분류
Table 2. The four HLL-line catagories

Category	HLL line	LLL instruction
1	DO REG(j) OUT ENDPROCEDURE	DO REG(j) OUT RET
2	PROGRAM np ENDIF PROCEDURE p ENDPROGRAM	No LLL instruction

3	WHILE x
	ELSE
	IF x
	ENDWHILE
		IF x ELSE ADR0
		(GO TO ADR+)

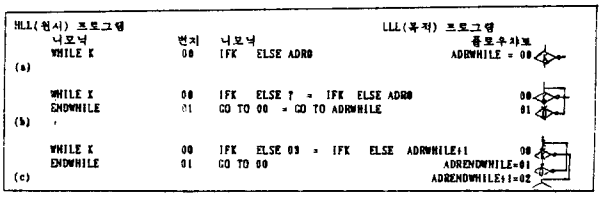
4	CALL p	CALL ADSP

표 3. IF.. ELSE.. ENDIF 컴파일 알고리즘
Table 3. Compilation algorithm for the IF.. ELSE.. ENDIF construction



a) First step : IF..
b) Second step : ELSE.. ; ADR0 = ADRLSE + 1
c) Third step : ENDIF ; ADR+ = ADREND + 1

표 4. WHILE.. ENDWHILE 의 컴파일 알고리즘
Table 4. Compilation algorithm for the WHILE.. ENDWHILE construction



a) First step : WHILE..
b) Second step : END WHILE ; ADRWHILE
c) Third step ; END WHILE ; ADR) = ADRENDWHILE + 1

5. 모의 실험 및 고찰

이 디지탈 시스템은 레지스터 값에 의해 움직이는 로보트 팔 , 재료 (물체) 그리고 그 재료에 레지스터의 값이 지정한 깊이 3 의 구멍을 내는 시스템으로 그림 5 에 보인다.

(S , T , U , V) 는 현재의 작업 수행상태를 나타내는데 여기서 S 는 로보트 팔의 평면좌표 , T 는 로보

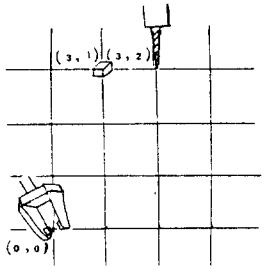


그림 5. 디지탈 시스템의 모형
Fig. 5. Digital system model

트 팔이 물체를 잡았는지의 여부 , U 는 물체의 평면좌표 , V 는 물체에 난 구멍의 깊이로 한다. 그림 6 이 문제해결을 위한 순서도이다.

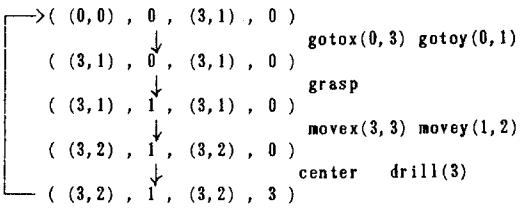


그림 6. 문제 해결을 위한 순서도
Fig. 6. The state diagram for solving problem

위 문제를 해결 하기위해 본 연구에서는 6개의 명령으로 나누었는데 다음과같다.

	A B C D E F G H I J K L M N O P
GOTOX(DE,FG)	0 0 0 D E F G
GOTOY(DE,FG)	0 0 1 D E F G
MOVEX(DE,FG)	0 1 0 D E F G
MOVEY(DE,FG)	0 1 1 D E F G
UNGRASP	1 0 0
GRASP	1 0 1
DRILL(DE)	1 1 0 D E
CENTER	1 1 1

명령 실행후 레지스터 값의 변화는 다음과 같다.

REGISTER[0] = x : x 만큼 X축으로 이동

REGISTER[1] = y : y 만큼 Y축으로 이동

REGISTER[2] = x : 물체를 로보트가 잡고
x 만큼 X축으로 이동

REGISTER[3] = y : 물체를 로보트가 잡고
y 만큼 Y축으로 이동

REGISTER[4] = 1 : 물체를 잡음을 의미
0 : 물체를 잡지 않음을 의미

REGISTER[5] = 1 : 물체에 구멍을 낼 중심과 DRILL의 중심이 일치함을 의미
= 0 : 물체에 구멍을 낼 중심과 DRILL의 중심이 일치하지 않음을 의미

REGISTER[6] = Z : 물체에 깊이 Z 로 구멍을 냄을 의미

각 명령은 그림 7 에서와 같은 논리함수로 표현된다.

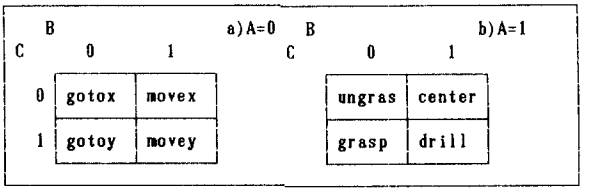


그림 7. 전체 시스템의 Karnaugh 도
Fig. 7. Karnaugh maps for instruction

부프로그램(gotox 명령)에 P-함수를 적용한 예이다.

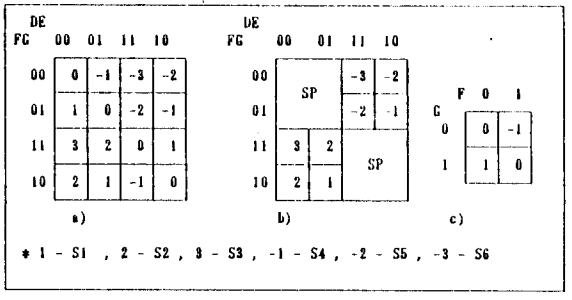


그림 8. gotox 명령의 Karnaugh 도
Fig. 8. Karnaugh maps for gotox instruction

Prime P^0-function :

$A_0 = \langle D'F' + DF ; SP \rangle$
$A_1 = \langle D'E\,F\,G' ; S1 \rangle$
$A_2 = \langle D'E\,F\,G + D'E'F'G' ; S2 \rangle$
$A_3 = \langle D'E'F\,G ; S3 \rangle$
$A_4 = \langle D\,E'F'G ; S4 \rangle$
$A_5 = \langle D\,E\,F'G + D\,E'F'G' ; S5 \rangle$
$A_6 = \langle D\,E\,F'G' ; S6 \rangle$

Prime P^1-function :

$B_0 = A_2 T_E A_1 = \langle D'F\,G' ; S2*E' + S1*E \rangle$
$B_1 = A_3 T_E A_2 = \langle D'F\,G ; S3*E' + S2*E \rangle$
$B_2 = A_4 T_E A_5 = \langle D\,F'G ; S4*E' + S5*E \rangle$

Prime P^2-function :

$C_0 = B_0 T_F B_1 = \langle D'F ; G'(S2*E' + S1*E) + G(S3*E' + S2*E) \rangle$
$C_1 = B_0 T_F B_2 = \langle D\,F' ; G'(S4*E' + S5*E) + G(S5*E' + S6*E) \rangle$

Prime P^3-function :

$D_0 = A_0 T_F C_0 = \langle D' ; F'*SP + F(G'(S2*E' + S1*E) + G(S3*E' + S2*E)) \rangle$
$D_1 = C_1 T_F A_0 = \langle D ; F'(G'(S4*E' + S5*E) + G(S5*E' + S6*E)) + F*SP \rangle$

Prime P -function :

$E_0 = D_0 T_0 D_1 = \langle 1 ; D'F'*SP + D'F (G'(S2*E' + S1*E) + G (S3*E' + S2*E)) + D F'(G'(S4*E' + S5*E) + G (S5*E' + S6*E)) + D F *SP \rangle$

위 과정에서 판정변수를 얻어서 최적 그래프를 얻을 수 있는데 그림 9 에 표시된다.

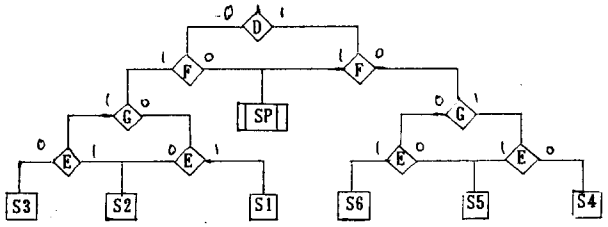


그림 9. gotox의 최적그래프
Fig. 9. Optimal graph for gotox

K신호를 시작 신호로 하여 프로그램한 것이 표 5이다. 모의 실험장치의 전체 프로그램을 실행한 각 명령 수행후의 결과는 그림 10과 같다.

표 5. gotox명령의 마이크로파스칼 프로그램과 하위레벨언어
Table 5. HLL(source) and LLL(object) program for gotox instruction

HLL(source) program		LLL(object) program	
ADR D	Mnemonics	ADR D	Mnemonics
00	PROGRAM 1		PROGRAM PP
01	WHILE K	00	IFK ELSE 02
02	ENDWHILE	01	GOTO 0
03	IF D	02	IFD ELSE 17
04	IF F	03	IFF ELSE 6
05	CALL SP0	04	CALL 30
06	ELSE	05	GOTO 16
07	IF G	06	IFG ELSE 12
08	IF E	07	IFE ELSE 10
09	DO REG(0) < 5	08	DOREG(0)<5
10	ELSE	09	GOTO 11
11	DO REG(0) < 4	10	DOREG(0)<4
12	ENDIF		ENDIF
13	ELSE	11	GOTO 16
14	IF E	12	IFE ELSE 15
15	DO REG(0) < 6	13	DOREG(0)<6
16	ELSE	14	GOTO 16
17	DO REG(0) < 5	15	DOREG(0)<5
18	ENDIF		ENDIF
19	ENDIF		ENDIF
20	ENDIF		ENDIF
21	ELSE	16	GOTO 30
22	IF F	17	IFF ELSE 29
23	IF G	18	IFG ELSE 24
24	IF E	19	IFE ELSE 22
25	DO REG(0) < 2	20	DOREG(0)<2
26	ELSE	21	GOTO 23
27	DO REG(0) < 3	22	DOREG(0)<3
28	ENDIF		ENDIF
29	ELSE	23	GOTO 28
30	IF E	24	IFE ELSE 27
31	DO REG(0) < 1	25	DOREG(0)<1
32	ELSE	26	GOTO 28
33	DO REG(0) < 2	27	DOREG(0)<2
34	ENDIF		ENDIF
35	ENDIF		ENDIF
36	ELSE	28	GOTO 38
37	CALL SP0	29	CALL 38
38	ENDIF		ENDIF
39	ENDIF		ENDIF
40	PROCEDURE SP0		SUBPROGRAM SP0
41	IF E	30	IFE ELSE 36
42	IF G	31	IFG ELSE 34
43	DO REG(0) < 0	32	DOREG(0)<0
44	ELSE	33	GOTO 35
45	DO REG(0) < 4	34	DOREG(0)<4
46	ENDIF		ENDIF
47	ELSE	35	GOTO 40
48	IF G	36	IFG ELSE 39
49	DO REG(0) < 1	37	DOREG(0)<1
50	ELSE	38	GOTO 40
51	DO REG(0) < 0	39	DOREG(0)<0
52	ENDIF		ENDIF
53	ENDIF		ENDIF
54	ENDPROCEDURE	40	RET
55	ENDPROGRAM	41	ENDPROGRAM

Modeling Test Unit

```
VARM[0..F] = 0000011000000000
RUN

REGISTER[0] = 3  REGISTER[1] = 0  REGISTER[2] = 0
REGISTER[3] = 0  REGISTER[4] = 0  REGISTER[5] = 0
REGISTER[6] = 0  REGISTER[7] = 0
```

a) gotox 명령

```
VARM[0..F] = 0010001000000000
RUN

REGISTER[0] = 0  REGISTER[1] = 1  REGISTER[2] = 0
REGISTER[3] = 0  REGISTER[4] = 0  REGISTER[5] = 0
REGISTER[6] = 0  REGISTER[7] = 0
```

b) gotoy 명령

```
VARM[0..F] = 1010000000000000
RUN

REGISTER[0] = 0  REGISTER[1] = 0  REGISTER[2] = 0
REGISTER[3] = 0  REGISTER[4] = 1  REGISTER[5] = 0
REGISTER[6] = 0  REGISTER[7] = 0
```

c) grasp 명령

```
VARM[0..F] = 0101111000000000
RUN

REGISTER[0] = 0  REGISTER[1] = 0  REGISTER[2] = 0
REGISTER[3] = 0  REGISTER[4] = 0  REGISTER[5] = 0
REGISTER[6] = 0  REGISTER[7] = 0
```

d) movex 명령

```
VARM[0..F] = 0110110000000000
RUN

REGISTER[0] = 0  REGISTER[1] = 0  REGISTER[2] = 0
REGISTER[3] = 1  REGISTER[4] = 0  REGISTER[5] = 0
REGISTER[6] = 0  REGISTER[7] = 0
```

e) movey 명령

```
VARM[0..F] = 1110000000000000
RUN

REGISTER[0] = 0  REGISTER[1] = 0  REGISTER[2] = 0
REGISTER[3] = 0  REGISTER[4] = 0  REGISTER[5] = 1
REGISTER[6] = 0  REGISTER[7] = 0
```

f) center 명령

```
VARM[0..F] = 1101100000000000
RUN

REGISTER[0] = 0  REGISTER[1] = 0  REGISTER[2] = 0
REGISTER[3] = 0  REGISTER[4] = 0  REGISTER[5] = 0
REGISTER[6] = 3  REGISTER[7] = 0
```

g) drill 명령

그림 10. 수행 결과
Fig. 10. Execution result

그림 10에서 볼 수 있는 것처럼 gotox(0,3)후 레지스터 R[0]의 값이 3으로 변화하였으므로 결과적으로 로보트 팔을 X축 상으로 3 만큼 이동될 수 있음을 보였고, gotoy(0,1)명령 수행후는 레지스터 R[1]의 값이 1 로 변화되었으며, grasp명령 수행에는 레지스터 R[4]의 값이 1로, movex(3,3)명령 수행 후에는 레지스터 R[2]의 값이 0 로 , movey(1,2)명령 수행 후에는 레지스터 R[3]의 값이 1 로 , center명령 후에는 레지스터 R[5]의 값이 1 로 , drill(3)명령 후에는 레지스터 R[6]의 값이 3 으로 된다.

6. 결 론

본 논문에서는 순차제어시스템의 체계적 상태머신(algorithmic state machine)에 의한 모델화, P-함수 도입, 마이크로파스칼을 사용한 프로그램 실현, 컴파일러 개발로 다음과 같은 결론을 얻게 되었다.

첫째, 디지타 제어 시스템 프로그램의 신뢰성을 개선하였으며, 특히 마이크로파스칼과 같은 구조적 고급언어(structured high-level-language)로 분기명령(branch instruction)과 호출명령(call instruction)의 번지를 정확하게 계산할 수 있게 되었다.

둘째, 기본적으로 하드웨어 개념과 소프트웨어 개념을 도입하여 휨웨어 개념을 제공하였다. 이것으로 인해 디지탈 시스템의 설계에서 항상 내재한 실험작업의 어려움을 쉽게 해결하도록 했다.

셋째, 프로그램의 이식성을 들 수 있는데 이것은 컴퓨터 시스템에 따라 새로 프로그램을 할 필요없이 컴파일러만을 따로 구성하면 되도록 간단해 졌다.

넷째, 어떤 순차제어(Seqence Control)도 체계적인 방법으로 이룰 수 있다는 가능성을 제시하였다.

결론으로 실제시스템을 모델화하여 분석하는 일이 간단해졌으며 고급언어사용으로 인터페이스만 가능하면 디지탈 시스템의 순차제어를 효율화 할 수 있게 되었다.

참 고 문 헌

(1) Daniel A. Mange, "A High-Level-Language Programmable Controller," ; "Part I:A Controller for Structured Microprogramming," ;"Part II:Microcompilation of the High-Level Language Micropascal" IEEE Micro, 1986, Feb. pp.25-41, April. pp.47-63

(3) A. Thyase,"P-Functions: A new tool for the analysis and synthesis of binary programs," IEEE Trans. Comput. vol. C-30, no. 2, pp. 126-134, 1981, Feb.

(4) S. B. Akers, "Binary decision diagrams,"IEEE Trans. Computers, Vol. C-27, No.6, 1978, June

(5) Fredrick J.Hill and Gerald R.Peterson ,"Digital Systems : Hardware Organization and Design ", 2nd ed. John Wiley & Sons, Inc. 1978

(6) Milos D. Erdegovac ,"Digital Systems and Hardware/Firmware Algorithms", John Wiley & Sons, Inc. 1985

(7) 김 복기 "마이크로파스칼에 의한 순차제어시스템의 프로그램 개발" 연세대학교 석사논문. 1986