

New Test System

Micro Compiler Users Manual

Revision 0.4

**Revision History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rev | Date | Description | Revised By | Approval |
| 0.0 | 2019.04.26 | 신규 작성 | Ito |  |
| 0.1 | 2019.07.29 | Modified Rv items | Ito |  |
| 0.2 | 2019.8.30 | TPM,DBFORM관련수정 | Ito |  |
| 0.3 | 2019.10.24 | 지적사항및사양협의내용수정 | Ito |  |
| 0.4 | 2020.04.28 | 전체검토 | Ito |  |
|  |  |  |  |  |

Copyright © 2019 주식회사 헬릭스

본 Manual 내용의 전부 또는 일부를 당사의 허가 없이 무단 전재 또는 복제하는 것을 금지합니다.

**목차**

[1. Overview 5](#_Toc477617354)

[2. Micro Program Register 5](#_Toc477617355)

[2.1 Sequence Register List 5](#_Toc477617356)

[2.2 Sequence Register Description 8](#_Toc477617357)

[2.2.1 PCSTRADR 8](#_Toc477617358)

[2.2.2 PCINTADR 9](#_Toc477617359)

[2.2.3 PCSYNC 10](#_Toc477617360)

[2.2.4 WAIT 10](#_Toc477617361)

[2.2.5 CONFLG 10](#_Toc477617362)

[2.2.6 LDIm 11](#_Toc477617363)

[2.2.7 REFTM 11](#_Toc477617364)

[2.2.8 PAUTM 11](#_Toc477617365)

[2.2.9 TRATM 11](#_Toc477617366)

[2.2.10 TMRTM 12](#_Toc477617367)

[2.2.11 TIMDm 12](#_Toc477617368)

[2.2.12 REFEN 12](#_Toc477617369)

[3. Micro Program Construction 13](#_Toc477617370)

[3.1 Sequence Command List 13](#_Toc477617371)

[3.1.1 Main Command 13](#_Toc477617372)

[3.1.2 Sub Command 14](#_Toc477617373)

[3.2 Main Command Description 15](#_Toc477617374)

[3.2.1 NOP 15](#_Toc477617375)

[3.2.2 STPS 16](#_Toc477617376)

[3.2.3 WAIT 17](#_Toc477617377)

[3.2.4 RET 18](#_Toc477617378)

[3.2.5 JUMP 19](#_Toc477617379)

[3.2.6 JSR 20](#_Toc477617380)

[3.2.7 RTN 21](#_Toc477617381)

[3.2.8 JMFLG 22](#_Toc477617382)

[3.2.9 JMFLG 23](#_Toc477617383)

[3.2.10 JEXFLG 24](#_Toc477617384)

[3.2.11 JPASS 25](#_Toc477617385)

[3.2.12 JFAIL 26](#_Toc477617386)

[3.2.13 JETn 27](#_Toc477617387)

[3.2.14 JNC 28](#_Toc477617388)

[3.2.15 JNIn 29](#_Toc477617389)

[3.2.16 JNINDm 30](#_Toc477617390)

[3.2.17 JNInLDm 31](#_Toc477617391)

[3.2.18 VSBMPLEV 32](#_Toc477617392)

[3.2.19 VSBMP 33](#_Toc477617393)

[3.2.20 IOBMPLEV 34](#_Toc477617394)

[3.2.21 IOBMP 35](#_Toc477617395)

[3.3 Sub Command Description 36](#_Toc477617396)

[3.3.1 LDIm Set 36](#_Toc477617397)

[3.3.2 TIMDn Set 37](#_Toc477617398)

[3.3.3 REFTM Set 38](#_Toc477617399)

[3.3.4 PAUSE Set 39](#_Toc477617400)

[3.3.5 TRA Set 40](#_Toc477617401)

[3.3.6 TIMER Set 41](#_Toc477617402)

[3.3.7 REFST 42](#_Toc477617403)

[3.3.8 PAUST 43](#_Toc477617404)

[3.3.9 TRAST 44](#_Toc477617405)

[3.3.10 TMRST 45](#_Toc477617406)

[3.3.11 INTDS 46](#_Toc477617407)

[3.3.12 EXINT 47](#_Toc477617408)

# Overview

본 Test System에서는 ALPG(algorithmic pattern generator)에서 Test Pattern을 발생시킨다.

Test Pattern 발생을 제어하는 Program을 Micro Pattern Program이라 한다.

아래에 Compiler의 Input, Output을 개략 도와 본문의 구성에 대해 간단하게 설명하였습니다.

▶ Micro Pattern Compiler 의 Input, Output

Object File(\*.pat)

Pattern Program Text File(\*.asc)

Intermediate File(\*.mif)

Define List File(\*.mdf)

Error File(\*.mer)

patcomp

(Option 선택에 따라서 Intermediate File과 Define File의 생성 유무를 결정 할 수 있습니다.)

# Micro Program 설명

Micro Pattern Program은 Micro Pattern을 기술한 언어로써 Device Test을 목적으로 하고 있습니다. Micro Program의 이해를 돕기 위해 다음과 같은 보충 설명을 하였습니다.

Pattern Program의 전체 구성과 문법에 대해 간략하게 기술 하였습니다.

* + **Micro Pattern Program의 구성**

|  |
| --- |
| 1)Program의 시작  MPAT문  •  •  2)Common : DEFINE PART  •  3)PATTERN PROGRAM의 시작  START문  •  4)Common :  PATTERN PROGRAM PART  •  •  5)MODULE의 시작  MODULE BEGIN문  •  •  6)Module : DEFINE PART  •  PATTERN PROGRAM의 시작  START문  •  8)Module :  PATTERN PROGRAM PART  •  •  9)MODULE의 끝  MODULE END문  MODULE BEGIN문  •  Module part는 32개까지  사용이 가능 합니다.  •  MODULE END문  •  •  •  11)Program의 종료  END문 |

▶MPAT문

PROGRAM의 시작 부분에 기술됩니다.

Object File name을 지정（선택적）.

Object File name을 생략하면 Source File name으로 작성된다.

▶COMMON: DEFINE PART

COMMON PART의 RDX, IL\_MODE, MODE, REGISTER문 등이 기술됩니다. 모두 생략이 가능 하며 생략 시 Default 값이 Set됩니다. Common Part에 기술된 Define문들은 Module Part의 default가 됩니다.

▶START문

시작 ADDRESS LINE NUMBER를 기술합니다. START문은 PATTERN PROGRAM PART를 기술하기 바로 전에 꼭 기술 하여야 합니다.

▶COMMON: PATTERN PROGRAM PART

ALPG의 SEQUENCE 제어명령과 ALU 명령을 기술합니다.

▶MODULE BEGIN문

Module part의 시작부분에 기술이 됩니다.

Module문은 16개까지 기술 가능합니다.

▶MODULE: DEFINE PART

MODULE PART의 RDX, IL\_MODE, MODE, REGISTER문 등이 기술됩니다. 모두 생략이 가능 하며 생략 시 Default 값이 Set됩니다. MODULE Part에 기술된 Define문들은 Common Part define문들보다 우선합니다.

▶MODULE: PATTERN PROGRAM PART

ALPG의 SEQUENCE 제어명령과 ALU 명령을 기술합니다.

▶MODULE END문

Module part의 끝부분에 기술이 됩니다.

▶END문

PROGRAM의 마지막 부분에 기술됩니다.

* + **Micro Pattern Program의 문법**

한 행 최대 기술가능 문자는 512개입니다. Mnemonic과 Mnemonic사이의 다수의 Space 또는 Tab의 기술이 가능합니다. 각 구성요소의 설명과 기술 시 주의 할 점에 대한 자세한 설명을 아래에 기술 하였습니다.

▶문자

|  |  |  |
| --- | --- | --- |
| 분류 | 문자 | 비고 |
| 영문자 | A~Z,a~z | 소문자는 대문자로 바꿔  Compile 합니다. |
| 숫자 | 0~9 |  |
| 특수기호 | @ = , ; : % \_ - + \* ( ) # . / ^ < > & ‘ “ |  |

영문자의 대소문자는 모두 같은 것으로 처리됩니다. 예를 들어 TEST와 test은 같은 의미를 가지게 됩니다.

►특수 기호의 기능

|  |  |
| --- | --- |
| 기 호 | Description |
| @ (at mark) | Line Joint로써 현 Line의 기술을 다음 Line에 기술을 가능  하게 합니다.  @ 이후의 문자는 무시된다. |
| = (equal) | Register문에서 Register name 과 Value 사이에 ‘=’을 기술해야 합니다. 또한 Address Define문에서 X, Y 와 B, N, Z사이에 기술 하여야 합니다. |
| , (comma) | Address Define문에서 X, Y, Z, N, B구분자, Bank Load(Save) Address 에서 구분자로 사용됩니다. |
| ; (semicolon) | User가 Comment 작성할때 사용 합니다. |
| : (colon) | Label name의 마지막에 기술 하여야 한다 |
| % (percent) | IFN, IFE, ENDC, REMARK, INSERT 앞에 반드시 기술 하여야합니다. |
| \_ (under bar) | File name, Sdef name, Label name에 기술이 가능합니다. |
| - (minus) | ALU Mnemonic 에 기술되어 있고 Address Define, Bank Address 문에서 연속의 의미로써 기술 됩니다. 또한  Sequence Mnemonic에서 Address Line 을 지정 할 때 연산 기호로써 사용됩니다. |
| + (plus) | ALU Mnemonic에 기술 되고 Sequence Mnemonic에서 Address Line을 지정 할 때 연산기호로써 사용됩니다. |
| ~~\* (asterisk)~~  . (dot) | ALU Mnemonic에 기술 되고 Sequence Mnemonic에서 Address Line을 지정 할 때 현재 Line number의 의미로써 사용됩니다. |
| (, ) (parenthesis) | IF문의 연산 식에서 연산의 우선순위를 조정할 때 사용됩니다. |
| # (hash mark) | 16진수의 수를 기술할 때 사용됩니다. |
| . (point) | IF문의 연산 식에서 연산자의 양 옆에 기술 됩니다. |
| / (slash) | ALU Mnemonic에 기술 됩니다. |
| ^ (caret) | ALU Mnemonic에 기술 됩니다. |
| < (less than sign) | ALU Mnemonic에 기술 됩니다. 또한 MPAT문에서 |
| > (grater than sign) |  |
| & (ampersand) | ALU Mnemonic에 기술 됩니다. |
| ‘ (single quotation) | ALU Mnemonic에 기술 됩니다. |
| “ (double quotation) | ALU Mnemonic에 기술 됩니다. |

►진수에 따른 표기

|  |  |  |
| --- | --- | --- |
| 분류 | 표기 | 비고 |
| 10진수 | 0~9로 이루어진 정수 |  |
| 16진수 | 0~9, A~F로 이루어진 정수 |  |

▶Micro Pattern Address Line 기술 방법

Sequence 명령어에서 주소를 기술 할 때 사용합니다. 다음과 같은 기술 방법이 있습니다.

► .

현재 Address Line의 주소 지정입니다.

► \* +(-) n (n=1~5)

현재 Address Line을 기준으로 위, 아래로 5 Line 범위내의 주소를 지정합니다.

► Label name

Label이 기술된 Address Line의 주소 지정입니다.

► Label name +(-) n

Label이 기술된 Address Line을 기준으로 위, 아래로 5 Line 범위내의 주소를 지정합니다.

▶%IFN, IFE의 기술 방법

IFN과 IFE은 Compile 조건 문으로써 사용됩니다. 조건으로는 단일조건과 연산조건이 사용 됩니다. 여기서 사용되는 조건문자는 영문자(A~Z)만이 가능하며 10개까지 사용이 가능합니다.

▶’@’(At mark) :

‘@’(At mark)의 사용은 명령어의 기술을 다음Line에 계속할 수 있게 합니다. ‘@’(At mark)는 무한으로 사용가능 합니다.

▶Comment :

Comment는 ‘;’(Semicolon)으로 시작해서 ‘\n’(New Line character)로 끝이 납니다. Comment는 ‘@’에 의한 연속기술이 불가능 합니다.

▶%Remark :

Remark문은 최대 80문자까지 가능합니다. 또한 최초 Remark문만이 유효하며 2회 이상의 기술은 무시하게 됩니다.

▶’:’colon :

‘:’colon은 Label의 마지막에 기술된다

▶File name, Sdef name, Label name:

32문자까지 가능합니다. 사용가능 문자는 영대문자(A~Z), 숫자(0~9), 특수문자(\_)의 사용이 가능 합니다.

* + **제한 사항**

|  |  |  |  |
| --- | --- | --- | --- |
| no | 제한 사항 | Max | Remark |
| 1 | 한 Line 기술가능 문자 수 | 512문자 |  |
| 2 | File name, Label name, Sdef name 기술 가능 문자 수 | 32문자 | File name은 확장자를 제외한 문자 개수 입니다. |
| 3 | %Remark문에 기술 가능 문자 수 | 80문자 |  |
| 4 | Module 최대 기술가능 개수 | 32(EA) |  |
| 5 | IF문 겹 선언 최대 기술가능 개수 | 10(EA) |  |
| 6 | IF문을 위한 Condition Label 문자 수 | 10문자 |  |
| 7 | SDEF 기술 가능 개수 | 512(EA) |  |
| 8 | Start문의 최대 개수 | 64(EA) | 하나의 Module 중 최대 4개까지 사용가능 |
| 9 | %INSERT 기술 가능 개수 | 16(EA) |  |
| 10 | %INSERT 겹 선언 기술 가능 개수 | 5(EA) |  |

* + **Micro Pattern Program의 실행**

Micro Pattern Compiler 는 Text 환경(CUI)모두 실행이 가능 합니다.

* + - **CUI(Charcter User Interface)**

▶Description

Text Command 창에서 Compiler를 실행합니다. patcomp가 실행 명령어 이고

Parameters에 따라 결과물이 달라질 수 있습니다. Parameters 입력에 있어서 Pattern Program File name(\*.asc)의 입력은 필수 입니다.

▶Format :

% patcomp Options Source\_File\_Name

▶Parameters :

► Source File name : Compile이 필요한 Pattern Program Text File name

► -I : Intermediate File 생성 유무(이 Option이 없을 시 Intermediate File을 생성하지 않습니다.)

▶Example

% patcomp –I EXTEST.asc

**제어 명령**

▶ 제어 명령에 따라 Mode 설정, Opcode 및 data 설정, Micro bit 설정등이 결정 되어집니다. 따라서 제어 명령에 따른 알맞은 인자들을 사용해야 합니다. 본 장에서는 제어 명령 keyword와 알맞은 인자들의 기술에 대해 설명 하였습니다.

* + **MPAT문**

[Function]

Pattern Program name를 지정합니다.

PROGRAM의 시작 부분에 기술한다.

[Format]

MPAT program-name

►program-name (선택적)

Pattern Program 명칭입니다.

이 명칭은 Compile 후의 Object File명칭으로 사용되고 Test Plan Program에 두어 Pattern File을 지정할 때의 식별명이 됩니다.

영문 대문자(A~Z), 숫자(0~9), ‘\_’의 32 문자 이내에서 지정가능 합니다.

program-name을 생략하면, Object File Name은 source File된다.

[Memo]

[Description]

[Exercise]

* + **RDX문**

[Function]

Program중에 기술하는 정수의 기수를 정의합니다.

[Format]

RDX n

►n

10 : 10진수, 16 : 16진수

[Description]

이 명령은 생략 가능 합니다. 생략시 (Default)n=16으로 간주 되어집니다. 다만, 정수를 16진수로 기술하는 경우, 선두의 문자가 A~F일 때 선두에 “0”또는 “#”을 부가해 주세요

[Exercise]

* + **MODE문**

[Function]

Mode문은 ALPG의 동작 mode의 지정을 실시 합니다. START문 이전에 지정해 주세요.

[Format]

MODE mode1 [mode2 …..]

mode1 [mode2 …..] : REFRESH, TP2INV,의 연속 기술이 가능 합니 다. 연속 기술시에는 Space 로 구분하여 주십시오

[Description]

각 Type의 지정 가능한 동작 mode를 그림 4-1에 나타냅니다.

[Memo]

►Module부와 Common부의 양쪽 모두에 동작 mode를 설정했을 경우 양쪽 모두로 지정된 mode가 유효하게 됩니다. REFRESH mode를 Module부와 Common부에서 다른 동작 mode로 지정 했을 경우 Module부의 동작 mode가 유효하게 됩니다.

REFRESH

TP2INV

TPM

n

DBFORM

n

MODE

그림 4-1 지정 가능한 동작 mode

REFRESH

►REFRESH

범용 타이머(TREFRESH)와 Interrupt 기능에 의해 Auto refresh Test를 실시하는 mode 입니다

►TP2INV

TP2 Register의 Data에 대해서, ARIRAM(영역 반전), FP함수, DFLG에 의한 Data 반전 제어를 실시하는 mode의 선언입니다.

본 mode를 생략 했을 경우, TP2 Register의 Data는 제어 대상에서 제외 됩니다.

[Exercise]

* + **REGISTER문**

[Function]

REGISTER문은 ALPG의 각 Register의 초기 설정을 실시하는 문장입니다.

[Format]

Register명=value

►Register명

사용할 Register의 이름 입니다.

►Value

사용할 Register의 기술 가능 범위 입니다.

[Description]

REGISTER문은 ALPG의 각 Register의 초기 설정을 실시하는 문장입니다.

【 메모 】

Time Register 의 값을 입력할때는 반드시 Decimal number로 입력하시기 바랍니다. Rdx 문에서 Hexadecimal Number로 설정이 되어도 Compiler는 Time Value를 10진수로 인식 합니다.

Test Plan Program과 Test Pattern Program에 의해 설명 될 수 있습니다.

16 진수 작성하고 Compile error된다.

[Exercise]

* + **ADDRESS DEFINE문**

[Function]

ADDRESS DEFINE문은 TEST Pattern을 발생하는 경우에 사용 합니다.

[Format]

ADDRESS DEFINE

Xl-m = Zp-q, Nr-s, Bt-u

Yl-m = Zp-q, Nr-s, Bt-u

XY = XY .OR. ZN 또는 ZN(ZN는 ZNB와 기술할 수 있습니다.)

★표 3-3 l, m, p, q, r, s, t, u

|  |  |
| --- | --- |
| 지정 범위 | 지정 가능 범위 |
| l,m | 0~23 |
| p,g | 0~23 |
| r,s | 0~7 |
| t,u | 0~7 |

(p<q, r<s, t<u)

[Description]

공통 부에서 지정하지 않은 경우, 공통 부의 ADDRESS Data는 Disable입니다.

모듈 부에서 지정하지 않은 경우, 공통 부의 ADDRESS DEFINE문이 Default가 됩니다.

◆N, Z, B

DUT 에 인가되는 X0 ~ X23 및 Y0 ~ Y23의 각 Address의 Bit에는, ADDRESS DEFINE문에 의해서 지정된 N Register, Z Register와 B Register에 Store되고 있는 Address Data를 출력 할 수 있습니다.

Xl-m = Zp-q, Nr-s, Bt-u 또는 Yl-m = Zp-q, Nr-s, Bt-u로 지정된 Bit에 의한, X 또는 Y의 Address의 OR를 출력할지, 지정된 Z, N, B의 Address를 우선해 출력할지를 이하의 명령으로 지정 합니다.

►XY .OR. ZN(ZNB)

X 또는 YAddress와 N, Z, B의 Address를 OR로 출력한다.

►ZN(ZNB)

X 또는 Y Address를 무시하고 N, Z, B Address만을 출력한다.

XY = XY .OR. ZN(ZNB) 또는 ZN(ZNB)의 지정을 생략 했을 경우, X 또는 Y Address 와 N, Z, B의 Address 를 OR로 출력합니다.

[Exercise]

Ex1.

ADDRESS DEFINE

X4-7=Z0-3 ;X4-7에 Z0-3을 OR 하고 출력

Y8-9=Z8-9 ;Y8-9에 Z8-9를 OR 하고 출력

XY=XY.OR.ZN

Ex2.

ADDRESS DEFINE

X0-7=Z0-3, N0-3 ;X0-3에 Z0-3을, X4-7에 N0-3을 출력

Y12-15=Z12-13, N0-1 ;Y12, 13에 Z12, 13을, Y14, 15에 N0, 1을 출력

XY=ZN

Ex1.

ADDRESS DEFINE

X0-3= B1-4 ;X0-3에 B1-4를 OR 해 출력

Y14-17=B0-3 ;Y14-17에 B0-3을 OR 해 출력

XY=XY.OR.ZNB

* + **SDEF문**

[Function]

Address 발생 명령, Data 발생 명령을 임의의 Mnemonic으로 정의 합니다. Pattern Program에서 본 명령에 의해서 지정된 Mnemonic을 사용하게 될 것입니다.

[Format]

SDEF name = instructions

►name

영문자로 시작되는 영문자, 숫자,\_(Under bar)로 32 문자 이내에서 지정합니다.

512개까지 지정 가능 합니다.

►Instructions

Address 발생 명령 및 Data 발생 명령

[Description]

공통 부에서 지정하지 않은 경우, 공통 부의 ADDRESS Data는 Disable입니다.

모듈 부에서 지정하지 않은 경우, 공통 부의 ADDRESS DEFINE문이 Default가 됩니다.

[Exercise]

Ex1. INIT, INC에 명령을 정의하는 경우

SDEF INIT = XB<0 YB<0 TP<0

SDEF INC = XB<XB+1 YB<YB+1^BX

:

START #0

IDXI1 #6

ST0: JNI2 ST0 W INC FP1

ST1: JNI2 ST1 R INC FP1

JZD ST0

STPS

:

* + **START문**

[Function]

Test Pattern 발생 명령을 저장하는 START Address/Pattern Name를 정의하기 위한 문입니다.

[Format]

START <value> <Pattern Name>

►value

#0~#3FFD 이 범위 안에서 지정이 가능합니다.

►Pattern Name

Pattern Name을 지정한다.

[Description]

START문은 Program중에 128개까지 기술할 수 있습니다.

[Exercise]

* + **MODULE문**

[Function]

1개의 Pattern Program내에 복수의 Pattern을 기술 하기 위한 분할에 사용됩니다.

[Format]

MODULE BEGIN

Module의 시작 부분에 기술합니다.

MODULE END

Module의 끝부분에 기술합니다.

[Description]

Pattern Program에서는, 최초의 모듈 부 이전을 공통 부라고 부릅니다. 모듈 부와 공통 부에서 모두 지정이 있는 경우, 모듈 부의 지정이 우선됩니다.

공통 부로 지정에 있어, 모듈 부로 지정이 없는 것은, 공통 부의 지정이 모듈 부에 반영됩니다. 모듈 문은 최대32개까지 사용가능 합니다.

[Exercise]

* + **BANK SAVE ADDRESS문**

[Function]

복수의 BANK를 가지는 Memory Device를 Test하기 위해서 XASV, XALD명령으로 BANK마다 의 Row address를 Save, Load하는 기능이 있습니다.

BANK SAVE ADDRESS문으로 Row측의 Bank Address를 지정하고, XASV 명령을 지정한 Cycle의 Row Address(ALPG로부터의 X Address)를, Bank Address 마다 저장할 수 있습니다.

[Format]

BANK SAVE ADDRESS a0 [, a1, a2, a3]

►a0

Row측의 Bank Address Bit 0에 대응하는 Address를 선택

►a1

Row측의 Bank Address Bit 1에 대응하는 Address를 선택

►a2

Row측의 Bank Address Bit 2에 대응하는 Address를 선택

►a3

Row측의 Bank Address Bit 3에 대응하는 Address를 선택

[Description]

a0 ~ a3는, Row측의 Bank Address로 사용하는 X, Y Address의 Address Bit를 선택합니다. 지정 할 수 있는 Address Bit는 다음과 같습니다.

►X0 ~ X23, Y0 ~ Y23, 0

"0"은, Bank Address에 관계없이, Fix Low가 되는 Bit를 나타냅니다. (Address Bit를 선택해 없는 것을 나타내 보입니다.)

a1 ~ a3은 생략 할 수 있습니다. 이 때, 생략 한 Bit에 대한 Bank Address는 선택되지 않습니다. ("0"(을)를 선택한 것과 같게 됩니다.)

Ex1.

BANK SAVE ADDRESS X13,14

a0~ a3는 다음과 같이 지정할 수 있습니다.

►단독지정

Address Bit를 1개씩 Comma로 단락 지어 지정 합니다.

Ex2.

BANK SAVE ADDRESS X13,X14,0,0

►연속 지정

Address Bit를 hyphen(-)으로 이어 지정합니다.

승순, 내림차순의 연속 지정을 할 수 있습니다.

Xn-m (0≤n,m≤23)

Yn-m

Xn-Xm

Yn-Ym

Ex3.

BANK SAVE ADDRESS X12-X15

►단독 지정과 연속지정의 혼합

단독 지정과 연속 지정은 조합해 지정할 수 있습니다.

Ex4.

BANK SAVE ADDRESS X12-X14,0

BANK SAVE ADDRESS문은 공통 부 또는 모듈 부로 지정할 수 있습니다. 공통 부를 전송 할 때, BANK SAVE ADDRESS문의 지정이 없으면 Bank Address는 선택될 수 없습니다. (a0~a3에 “0”을 지정한 것과 같게 됩니다.) 공통 부의 지정이 있으면 공통 부의 지정이 Bank Address를 선택하게 됩니다.

모듈 부를 전송 할 때, 모듈 부의 BANK SAVE ADDRESS문의 지정이 없으면 공통 부의 지정으로 Bank Address를 선택합니다. 모듈 부의 지정이 있으면 모듈 부의 지정으로 Bank Address를 선택합니다.

[Memo]

►BANK LOAD ADDRESS 문으로 지정된 X,Y Address는 PRESCRAM부를 지나지 않는 ALPG로 부터의 Address가 선택됩니다.

[Exercise]

* + **BANK LOAD ADDRESS문**

[Function]

복수의 BANK를 가지는 Memory Device를 Test하기 위해서 XASV, XALD명령으로 BANK마다의 Row address를 Save, Load하는 기능이 있습니다.

BANK LOAD ADDRESS문으로 Colum측의 Bank Address를 지정하는 것으로, XASV 명령으로 Save한 Row Address를 XALD 명령을 지정한 Cycle로 Bank Address 마다 Load할 수 있습니다.

[Format]

BANK LOAD ADDRESS a0 [, a1, a2, a3]

►a0

Column측의 Bank Address Bit 0에 대응하는 Address를 선택

►a1

Column측의 Bank Address Bit 1에 대응하는 Address를 선택

►a2

Column측의 Bank Address Bit 2에 대응하는 Address를 선택

►a3

Column측의 Bank Address Bit 3에 대응하는 Address를 선택

[Description]

a0 ~ a3는, Column측의 Bank Address로 사용하는 X, Y Address의 Address Bit를 선택합니다. 지정할 수 있는 Address Bit는 다음과 같습니다.

►X0 ~ X23, Y0 ~ Y23, 0

"0"은, Bank Address에 관계없이, Fix Low가 되는 Bit를 나타냅니다. (Address Bit를 선택해 없는것을 나타내 보입니다.)

a1 ~ a3은 생략 할 수 있습니다. 이 때, 생략 한 Bit에 대한 Bank Address는 선택되지 않습니다. ("0"(을)를 선택한 것과 같게 됩니다.)

Ex1.

BANK LOAD ADDRESS X13,14

a0 ~ a3의 지정 방법과 공통 부 또는 모듈 부의 지정은, BANK SAVE ADDRESS문과 같습니다.

[Memo]

►BANK LOAD ADDRESS 문으로 지정된 X,Y Address는 PRESCRAM부를 지나지 않는 ALPG로 부터의 Address가 선택됩니다.

[Exercise]

* + **END문**

[Function]

Program의 번역을 종료시킵니다.

Program의 마지막에 반드시 기술해 주세요.

[Format]

END

Pattern Program의 끝부분에 기술 합니다.

[Description]

[Exercise]

* + **%REMARK문**

[Function]

%REMARK문 이후, Carriage return까지의 문자열을, Revision 정보로써 Object file에 출력합니다.

[Format]

%REMARK comment

comment : 사용자가 Revision 또는 기타 정보를 기술합니다.

[Description]

기술 가능한 문자 수는, 80 문자까지입니다.

본 Statement는, Program중에 1회만 기술할 수 있습니다.

[Memo]

►공통 부, 모듈 부의 어느 쪽에도 기술할 수 있습니다. 다만, START문 이후에 기술할 수 없습니다.

►Revision 정보가 80 문자를 넘었을 경우, 81 문자 이후는 무시됩니다.

►본Statement를 Program중에 2회 이상 기술했을 경우, 2번째 이후의%REMARK문은 무시됩니다.

[Exercise]

* + **%IFE문, IFN문, %ENDC문**

[Function]

조건부 번역을 실시하기 위한 문장입니다.

[Format]

조건 Switch의 Global Switch를 영문자로 지정했을 경우

%IFE X

조건 번역의 대상이 되는 문장의 군

%ENDC

%IFN X

조건 번역의 대상이 되는 문장의 군

%ENDC

조건 Switch의 Global Switch를 연산 식에서 지정했을 경우

%IFE 연산 식

조건 번역의 대상이 되는 문장의 군

%ENDC

%IFN 연산 식

조건 번역의 대상이 되는 문장의 군

%ENDC

► X

조건 Switch가 되는 Global Switch의 지정입니다. 영문자 1 문자로 지정해 주세요.

►연산 식

조건 Switch가 되는 Global Switch의 연산 식 지정입니다.

►연산 식의 종류(논리 연산)

조건 Switch가 되는 Global Switch의 지정에, 논리 연산의 연산 식의 기술이 가능하고. 연산자로서는, 다음의 것을 사용할 수 있습니다.

또, ( )에 의해 연산의 우선 순위를 지정할 수 있습니다.

|  |  |
| --- | --- |
| .OR. | 논리합 |
| .XOR. | 배타적 부정 |
| .AND. | 논리적 |
| .NOT | 부정 |

연산의 우선 순위

( ) → .NOT. → .AND. →.OR.

.XOR.

병렬의 부분은 같은 우선 순위를 가집니다.

**[Description]**

►Switch를 단체로 지정했을 경우

조건 Switch로서 지정되어 있는 Global Switch가, 번역 Command중으로 지정되어 있을 경우를 진, 지정되어 있지 않은 경우를 가짜로 해, 조건 번역을 실시합니다.

|  |
| --- |
| %IFE x  옳을 경우 번역합니다.  옳지 않을 경우 번역하지 않습니다.  옳을 경우 번역하지 않습니다.  옳을 않을 경우 번역합니다.  :  %ENDC  %IFN x  :  %ENDC |

►연산 식 지정의 경우

조건 Switch로서 연산 지정되어 있는 Global Switch의 연산 결과가 1의 경우를 진, 0의 경우를 가짜로 해, 조건 번역을 실시합니다.

Global Switch의 지정은, 번역 Command로 실시합니다.

|  |
| --- |
| %IFE 연산 식  옳을 경우 번역합니다.  옳지 않을 경우 번역하지 않습니다.  옳을 경우 번역하지 않습니다.  옳을 않을 경우 번역합니다.  :  %ENDC  %IFN 연산 식  :  %ENDC |

[Exercise]

Ex1.

patcomp –D=AF

|  |
| --- |
| %IFE F.AND.(A.OR.H)  번역합니다.  번역하지 않습니다.  :  %ENDC  %IFN F.AND.(A.OR.H)  :  %ENDC |

【 메모 】

► 조건 번역 명령문은 상자(nest)가 가능하고, 10중까지 실시할 수 있습니다.

|  |
| --- |
| %IFE A  %IFN C  %IFE F  3종목  2종목  1종목  %ENDC  %ENDC  %ENDC  상자 상태 |

► (%IFE문+%IFN문)와 %ENDC문의 수가 일치하지 않아도 괜찮습니다만, %ENDC문의 수가 많으면 Error가 발생합니다.

► 조건 번역 명령문에 의해서 번역되지 않는 상태가 되어 있을 때에도, 다음의 명령은 번역의 대상이 됩니다..

%IFE, %IFN, %ENDC, %INSERT

► 조건 번역 명령문은, 행의 좌단에 없으면 안 됩니다.

►사용 예

조건 번역 명령문에 의한 Pattern Program의 번역 부분

번역 Command

Pattern Program % patcomp % patcomp -C A % patcomp -C C %patcomp -C AC

MPAT 1

∫

%IFE A

∫

%ENDC

∫

%IFN C

∫

%ENDC

∫

%IFE A

∫

%IFN C

∫

%IFE A

∫

%ENDC

∫

%ENDC

∫

%ENDC

∫

END

번역되는 부분

번역되지 않는 부분

* + **%INSERT/INSERT문**

[Function]

번역시에 다른 원시 파일을 삽입시키고, 번역을 실시하게 하기 위한 번역 명령문 입니다.

[Format]

INSERT file\_name

%INSERT file\_name

►file\_name

삽입하는 원시 파일명의 지정입니다. 다음과 같이 지정해 주세요.

Ex1.

INSERT ABC001

ABC001는, 원시 파일명입니다.

[Description]

규정된 원시 파일이, 그 행의 부분에 삽입되어 번역을 합니다.

지정을 실시할 때, %INSERT문과 INSERT문이 있습니다만, %INSERT문은 조건 번역 명령문(%IFE,%IFN, %ENDC)의 영향을 받지 않습니다. 즉,%INSERT 문의 경우는, 조건 번역 명령문중 번역되지 않는 장소에서 만나도 원시 파일의 삽입은 행해집니다.

[Memo]

►상자(삽입한 원시 파일내에, INSERT문이 있는 것)는 5중까지 가능합니다.

►ASC AO를 번역했을 경우

ASC A0

|  |
| --- |
| MPAT AO  REGISTER  IDX1=#1  IDX2=#2  INSERT BO  IDX7=#7  START #0  NOP  STPS  END |

ASC B0

|  |
| --- |
| IDX3=#3  INSERT CO  IDX6=#6 |

ASC C0

|  |
| --- |
| IDX4=#4  IDX5=#5 |

►번역 결과

1행째 MPAT A0

2행째 REGISTER

3행째 IDX1=#1

4행째 IDX2=#2

5행째

6행째 IDX3=#3

7행째

8행째 IDX4=#4

9행째 IDX5=#5

10행째 IDX6=#6

11행째 IDX7=#7

12행째

13행째 START #0

14행째 NOP

15행째 STPS

16행째 END

►본문은, 16개까지 기술 가능합니다.

[Exercise]

# Micro Program Register

Micro Program 관련 Register는 크게 아래와 같이 3종류로 나누어 진다.

Sequence Control Register

Address Generator Register

Data Generator Register

Start Address 및 Sequence Loop 설정 그리고 Address , Data 발생을 위해 연산하기 위해서는

반드시 설정을 해야 한다.

ALPG 관련 각종 Register 값의 설정을 변경 하기 위해서는 아래와 같은 형식이 필요하다.

[Format]

ALPG Register Name = 정수값

[Excise]

IDX1 = #100 ---16진수 지정

DCMR = 1 ---10진수 지정

[Notice]

Function Test 도중 자동 연산되어 변경 되는 Register들은 설정하면 안된다.

**2진, 8진의 기술은 할 수 없다.**

## Sequence Register List

Sequence 제어용 Register는 Micro Program실행을 제어하기 위한 Register이다.

각 Register의 설정과 관련해서는 아래 표1-1 Sequence Control Register List를 참고 한다.

표 1-1 Sequence Control Register List

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kind | Register | Name & Function | Bits | Set |
| PC | PCSTRADR(PC) | Program Start Address Register  Micro Program내 START기술과 같은 기능입니다.  Program의 시작 Address를 지정합니다. | 14 | R/  W |
| PCINTADR  (ISP) | Refresh Start Address Register  Micro Program내 ISP기술과 같은 기능입니다.  Refresh Timer에 의한 Interrupt 발생시 Jump할  Address를 지정합니다. | 14 | R/  W |
| PCSYNC | Program Counter Sync Register  PC Address와 일치하는 시점에 Trigger 신호를 발생합니다. | 14 | R/  W |
| JMP | CONFLG  (CFLG) | Condition Flag Register  JNCn(n=1-16) 명령 시 해당 Bit가 1상태 시 Jump하고 0상태 시 다음 명령을 수행합니다. | 16 | R/  W |
| LOOP | ~~LDm~~  IDXm  (m=1~16) | ~~LDn~~ IDXIn Loop Counter Set Register  ~~LS 및 LNn~~ IDXIn에 Setting할 Loop값을 저장 합니다. | 32 | R/  W |
| TIMER | TIMER1 | Normal Timer에서 사용한다.  T1과 JET에서 사용된다. | Time | R/  W |
| TIMER2 | Normal Timer와 Refresh Timer에서 사용된다.  Normal Timer Mode에서는 T2와 JET2로 사용된다.  Auto Refresh Timer Mode에서는 T3에서 사용된다. | Time | R/  W |
| TIMDn  (n=1~4) | Timer Set Register  TIMED 및 TMn에 Setting할 Timer값을 저장합니다. | Time | R/  W |
|  |  | Time | R/  W |
|  |  | Time | R/  W |

[주의]

Setting 제약과 관련한 내용은 Detailed Description 내용을 참조 한다.

## Sequence Register Description

### PCSTRADR(PC)

[Function]

Pattern을 개시할 Instruction Memory의 Address를 지정 합니다.

[Format]

PC = n

n 정수 0 ~ 4095 Program Start Address

[Description]

Micro Program의 수행을 Control하는 Sequence의 시작 Address를 지정하는 것으로

Pattern 내의 START와 동일한 기능을 합니다.

서로 다른 Address를 지시하고 있을 때는 나중에 설정된 Address가 지정됩니다.

[Exercise]

MPAT

START #0

NOP -- PC=0

MOOP -- PC=1

NOP -- PC=2

NOP -- PC=3 최종 Start Address

NOP -- PC=4

.

STOP

END

MAIN

LOAD\_MAPT(“PAT01");

TEST\_FUN(3, FAILSTOP); -- Start Address가 설정된다.

### PCINTADR(ISP)

[Function]

Refresh Interrupt 발생시 Jump할 Instruction Memory의 Address를 지정 합니다.

[Format]

ISP = n

n 정수 0 ~ 4095 Interrupt Routine Start Address

[Description]

T3 명령에 의해 TIMER2에서 지정된 시간이 경과 후 Refresh Interrupt가 발생하여 처리하는 Address로 Jump하도록되어 Pattern의 <INTADR>와 동일한 기능이다.

이때 서로 다른 Address를 지시하고 있을 때는 나중에 설정된 Address가 지정됩니다.

[Exercise]

### PCSYNC

[Function]

Data Log 취득 및 파형 확인을 위한 PC Triggering을 지정합니다.

[Format]

PCSYNC = n

n 정수 0 ~ 4095 PC Address

[Description]

PCSYNC Register는 1개가 존재하며 지정한 Address와 진행중인 Test Pattern의

Program Counter가 일치 시 Trigger 신호가 발생합니다.

[Exercise]

### WAIT

[Function]

진행중인 Test Pattern을 일시 정지 할수 있도록 지정 합니다.

[Format]

WAIT = n

n 정수 0 ~ 1 0 : 해제(Increment) , 1 : 설정(Wait)

[Description]

[Exercise]

### CONFLG

[Function]

JNCn Jump 명령 수행시 참조할수 있도록 값을 지정 합니다.

[Format]

CFLG = n

n 정수 0 ~ $F 0 : 해제(Increment) , 1 : 설정(Jump)

[Description]

[Exercise]

### IDXm

[Function]

JNIn 명령 실행시 참조하는 Loop 값을 지정한다.

[Format]

IDXm = n

m 정수 1 ~ 16 Loop값 저장 Register Number

n 정수 2 ~ $FFFFFFFF Loop 설정 값

[Description]

[Exercise]

### TIMER1

[Function]

T1에서 사용하는 Timer 값을 지정한다.  
JET에서 사용된다.

[Format]

TIMER1 = n

n 정수 100ns ~ 409.6s (1Rate ~ 4096Rate) Timer 설정 값

[Description]

[Exercise]

### TIMER2

[Function]

T2에서 사용하는 Timer 값을 설정한다.  
Normal Timer Mode에서는 JET2로 사용된다.

[Format]

TIMER2 = n

n 정수 100ns ~ 409.6s (1Rate ~ 4096Rate) Timer 설정 값

[Description]

[Exercise]

### TIMDm

[Function]

TIMER1 또는 TIMER2 명령 실행시 Timer 값을 지정한다.

[Format]

TIMDm = n

m 정수 1 ~ 4 Timer의 값을 저장 Register Number

n 정수 100ns ~ 409.6s (1Rate ~ 4096Rate) Timer 설정 값

[Description]

[Exercise]

### TIMDm

[Function]

REFST 또는PAUST,RASST,TMRST명령 수행시 Timer 값을 지정 합니다.

[Format]

TIMDm = n

m 정수 1 ~ 4 Timer값 저장 Register Number

n 정수 100ns ~ 409.6s (1Rate ~ 4096Rate) Timer 설정 값

[Description]

[Exercise]

# Micro Program Construction

## Sequence Command List

Sequence Control은 Micro Pattern의 기술에 의해 수행된다.

Sequence Control은 Main Command와 Sub Command로 구성되며 Program Counter를 제어하여

Instruction Memory에 저장된 Micro Pattern 구문을 수행한다.

### Main Command

|  |  |  |  |
| --- | --- | --- | --- |
| 구분 | Command | 내용 | 참조 |
| 기본  명령 | NOP | PC Increment를 수행한다. |  |
| STPS | Test Pattern을 종료 한다.(PASS) |  |
| STFL | Test Pattern을 종료 한다.(FAIL) |  |
| WAIT | Test Pattern을 Holding 한다. |  |
| RTN3 | Interrupt Routine에서 Return한다. |  |
| 무조건  Jump | JMP | PC를 해당 Address로 Jump한다. |  |
| JSR | PC를 Sub Routine Address로 Jump 한다. |  |
| RTN | Sub Routine Return을 수행한다. |  |
| 조건  Jump | JZD | 1 회 무조건 JUMP 후 2 번째 Data Inverting 신호를 발생하여 Data Inverting 후 3 번째 Increment를 실행한다. |  |
| JNC | Condition Flag에 설정된 값이 1이면 JUMP을 실행하고 0이면 Increment를 실행한다. |  |
| JEXFLGn | External Flag에Setting된 값이 1이면 Jump를 수행하고  0이면 Increment를 수행한다. |  |
| JMFLG | If JMFLG-Flag=1 then Jump else NOOP |  |
| JLML | If LM<>0 or JMFLG-Flag=0 then Jump else NOOP |  |
|  | JLMN | If m<>0 or JMFLG-Flag=0 then Jump else NOOP |  |
| Timer  Jump | JET2 | TIMER2 의해 실행 된 시간  Interrupt가 발생하기 전까지 Loop를 실행한다. |  |
| JET | TIMER1 의해 실행 된 시간 Flag가 발생하기 전까지 Loop를 실행한다. |  |
| Loop | IDXIn | IDXIn Register에 Setting된 값을 .LS Register에  Load후 횟수 만큼 Loop를 수행한다. |  |
| ~~JNC~~ | ~~IDXn Register에 Setting된 횟수 만큼 Loop를 수행한다.~~ |  |
| JNIn | IDXn Register에 Setting된 횟수 만큼 Loop를 수행한다. |  |
| JNInI | IDXn Register에 Setting된 횟수 만큼 Loop를 수행한다. |  |
| JNInD | IDXn Register에 Setting된 횟수 만큼 Loop를 수행한다. |  |
| Bump | VSBMP  LEVn Set | DPS Bump Level을 설정한다.  Data부에서 각 Level에 대한 전압값은 미리 Set해둔다. |  |
| VSBMP | DPS Bump Power On 명령을 실행한다. |  |
| IOBMP  LEVn Set | IO Bump Level을 설정한다.  Data부에서 각 Level에 대한 전압값은 미리 Set해둔다. |  |
| IOBMP | IO Bump Power On 명령을 실행한다. |  |

### Sub Command

|  |  |  |  |
| --- | --- | --- | --- |
| 구분 | Command | 내용 | 참조 |
| Loop | IDXIm Set | LDIm에 Loop Counter의 값을 설정한다. |  |
| Timer  Jump | TIMDn  Set | TIMDn에 Timer 기동 시간을 설정한다. |  |
| T3 | TIMER2 (Refresh)을 시작한다. |  |
| T1 | TIMER1(Pause)을 시작한다. |  |
| T2 | TIMER2(Normal)을 시작한다. |  |
| INTDS | Refresh Timer의 Interrupt 처리를 일시 Disable한다. |  |
| INTDS | Refresh Timer의 Interrupt처리를 일시 Disable 한다. |  |
| 조건  Jump | EXINTn | EXINTn External Interrupt를 설정 한다. |  |

## Main Command Description

### NOP

[Function]

기술된 Sequence를 수행후 그다음 Sequence로 진행한다.

[Format]

NOP

[Description]

Sequence Control Main 기본 명령이다.

NOP가 기술되면 자신을 한번 수행하고 다음 Sequence로 진행한다.

Program Counter의 값을 1 증가 시킨다.

**[Excise]**

|  |
| --- |
| MICRO1  <START>  **NOP** -- PC=0 자신을 실행하고 PC 1을 증가 시키고 다음 Sequence로 진행  **NOP** -- PC=1 자신을 실행하고 PC 1을 증가 시키고 다음 Sequence로 진행  **NOP** -- PC=2 자신을 실행하고 PC 1을 증가 시키고 다음 Sequence로 진행  **NOP** -- PC=3 자신을 실행하고 PC 1을 증가 시키고 다음 Sequence로 진행  **NOP** -- PC=4 자신을 실행하고 PC 1을 증가 시키고 다음 Sequence로 진행  STPS　-- PC=5  END |

### STPS(PASS)

[Function]

기술된 Sequence를 수행후 Sequence를 종료 한다.

[Format]

STPS

[Description]

Sequence Control Main 기본 명령이다.

STPS이 기술되면 자신을 한번 수행하고 Sequence를 종료 하여 Function Test를 완료한다.

**[Excise]**

|  |
| --- |
| MICRO1  <START>  NOP -- PC=0  NOP -- PC=1  NOP -- PC=2  NOP -- PC=3  NOP -- PC=4  **STPS**-- PC=5자신을 실행하고 Pattern을 종료  END |

### STFL(FAIL)

[Function]

기술된 Sequence를 수행후 Sequence를 종료 한다.

[Format]

STFL

[Description]

Sequence Control Main 기본 명령이다.

STFL이 기술되면 자신을 한번 수행하고 Sequence를 종료 하여 Function Test를 완료한다.

**[Excise]**

|  |
| --- |
| MICRO1  <START>  NOP -- PC=0  NOP -- PC=1  NOP -- PC=2  NOP -- PC=3  NOP -- PC=4  **STFL**-- PC=5자신을 실행하고 Pattern을 종료  END |

### WAIT

[Function]

기술된 Sequence를 수행후 참조 Register에 따라 정지 또는 다음 Sequence로 진행한다.

[Format]

WAIT

[Description]

Sequence Control Main 기본 명령이다.

Test 진행중인 특정 시점에서의 각 상태 Register의 확인을 위해 사용할수 있다.

Set 상태에서 계속 유지 이기 때문에 **해제를 하여 다음 명령을 수행하도록 하거나**

**ALPG STPS을 주어 강제 종료, 또는 새로운 Pattern을 수행할수 있도록 ALPG START**

명령을 주어야 한다.

위의 조치가 없으면 Pattern은 **무한 유지**를 한다.

**[Excise]**

|  |
| --- |
| ALPG  PAT10  MICRO1  <START>  NOP -- PC=0  NOP -- PC=1  **WAIT** -- PC=2 자신을 실행하고 WAIT 설정 상태에서는 계속 정지 상태  WAIT 해제 상태에서는 다음 명령 수행  NOP -- PC=3  NOP -- PC=4  STPS -- PC=5  END |

### RTN3

[Function]

작성된 Sequence를 실행 후 Interrupt Routine을 빠져 나올.

[Format]

RTN3

[Description]

Sequence Control Main 기본 명령이다.

RTN3가 작성되면이 줄을 실행 Interrupt Routine을 빠져 나가INTSTK에 저장된 Sequence Routine에 복귀한다.

복귀 지점은 발생 지점의 명령에 따라 (Jmp , Hold , Increment , Stack Address) 달라진다.

**[Excise]**

|  |
| --- |
| START  NOP TIMER2=64ms T2 -- PC=0  JNI3LD5 0 -- PC=1  NOP -- PC=2 Interrupt 발생 지점  NOP -- PC=3 RET　명령에 대한 복귀 지점  STPS  <INTADR>  NOP -- PC=4 Interrupt 발생에 대한 목적 지점  NOP -- PC=5  **RTN3** -- PC=6 자신을 실행하고 Interrupt 발생지점 다음으로 Return  END |

### JMP

[Function]

기술된 Sequence를 수행후 설정된 Address 무조건 Jump한다.

[Format]

JMP x

x \* , 수치 , Label , \*±수치 , Label±수치 Jmp Address (0 ~ 4095)

[Description]

Sequence Control Main 무조건 Jmp 명령이다.

조건 없이 Pattern에 기술된 Address로 Jmp한다.

**[Excise]**

|  |
| --- |
| START  NOP -- PC=0  NOP -- PC=1  **JMP A** -- PC=2 자신을 실행하고 무조건 A Lable로 Jmp  NOP -- PC=3  A:NOP -- PC=4 JMP명령에 대한 목적 지점  STPS -- PC=5  END |

### JSR

[Function]

기술된 Sequence를 수행후 설정된 Address 무조건 Jump한다.

[Format]

JSR x

x \* , 수치 , Label , \*±수치 , Label±수치 Jump Address (0 ~ 4095)

[Description]

Sequence Control Main 무조건 Jump 명령이다.

조건 없이 Pattern에 기술된 Address로 Jump한다.

**[Excise]**

|  |
| --- |
| START  NOP -- PC=0  **JSR A** -- PC=1 자신을 실행하고 A Lable로 무조건 Jump  　NOP -- PC=2 1 RTN명령에 대한 복귀 지점 1  STPS -- PC=3  B: NOP -- PC=4 2 JSR 명령에 대한 목적 지점  **RTN** -- PC=5 자신을 실행하고 2 JSR 다음 명령으로 Return  NOP -- PC=6  A: NOP -- PC=7 1 JSR 명령에 대한 목적 지점  NOP -- PC=8  **JSR B** -- PC=9 자신을 실행하고 B Lable로 무조건 Jump  　NOP -- PC=A 2 RTN명령에 대한 복귀 지점 2  NOP -- PC=B  **RTN** -- PC=C 자신을 실행하고 1 JSR 다음 명령으로 Return  END |

### RTN

[Function]

기술된 Sequence를 수행후 JSR 발생 다음 명령으로 복귀한다.

[Format]

RTN

[Description]

Sequence Control Main 무조건 Jump 명령이다.

자세한 사항은Command Description의 JSR를 참조 한다.

### IDXIn

[Function]

기술된 Sequence를 지정 횟수 수행후 Loop End Signal에 따라 Loop 또는 다음 Sequence로 진행한다.

[Format]

IDXIn j

j 정수 2 ~ $FFFFFFFF Loop Counter Direct 설정

n 정수 1~16 IDXn Counter 설정

[Description]

Sequence Control Main Loop 명령이다.

IDX1 ~ IDX16 Loop Counter에 대해 Direct Value를 지정하여 실행 한다.

설정한 횟수 만큼 반복 실행후 Loop End Signal을 받아 다음 명령을 수행한다.

**[Excise]**

|  |
| --- |
| START  NOP -- PC=0  NOP -- PC=1  NOP -- PC=2  NOP -- PC=3  **IDXI1 5**  -- PC=4 자신을 실행하고 5+2회째 까지 Loop 반복 하고  5+2회째 다음 명령 수행  NOP -- PC=5  NOP -- PC=6  STPS -- PC=7  END |

### JNIn

[Function]

기술된 Sequence를 수행후 Loop End Signal에 따라 Loop 또는 다음 Sequence로 진행한다.

|  |
| --- |
|  |

[Format]

JNIn x

n 정수 1~16 IDXn Counter 설정

x \* , 수치 , Label , \*±수치 , Label±수치 Jump Address (0 ~ 4095)

[Description]

Sequence Control Main Loop 명령이다.

IDX1 ~ IDX16 Loop Counter에 대해 Direct Value를 지정하여 실행 한다.

설정한 횟수 만큼 반복 실행후 Loop End Signal을 받아 다음 명령을 수행한다.

**[Excise]**

|  |
| --- |
| IDX1=5  IDX2=6  START  NOP -- PC=0  A: NOP -- PC=1  B: NOP -- PC=2  NOP -- PC=3  **JNI1 A** -- PC=4 자신을 실행하고 5+1회째 까지 Loop 반복 하고  5+2회째 다음 명령 수행  NOP -- PC=5  **JNI2 B** -- PC=6 자신을 실행하고 6+1회째 까지 Loop 반복 하고  6+2회째 다음 명령 수행  NOP -- PC=7  STPS -- PC=8  END |

### JNInI

[Function]

기술된 Sequence를 수행후 Loop End Signal에 따라 Loop 또는 다음 Sequence로 진행한다.

[Format]

JNIn x

n 정수 1~16 IDXn Counter 설정

x \* , 수치 , Label , \*±수치 , Label±수치 Jump Address (0 ~ 4095)

[Description]

Sequence Control Main Loop 명령이다.

IDX1 ~ IDX16 Loop Counter에 대해 Direct Value를 지정하여 실행 한다.

설정한 횟수 만큼 반복 실행후 Loop End Signal을 받아 다음 명령을 수행한다.

**[Excise]**

|  |
| --- |
| IDX1=5  IDX2=6  START  NOP -- PC=0  A: NOP -- PC=1  B: NOP -- PC=2  NOP -- PC=3  **JNI1 A** -- PC=4 자신을 실행하고 5+1회째 까지 Loop 반복 하고  5+2회째 다음 명령 수행  NOP -- PC=5  **JNI2 B** -- PC=6 자신을 실행하고 6+1회째 까지 Loop 반복 하고  6+2회째 다음 명령 수행  NOP -- PC=7  STPS -- PC=8  END |

### JNInD

[Function]

기술된 Sequence를 수행후 Loop End Signal에 따라 Loop 또는 다음 Sequence로 진행한다.

[Format]

JNIn x

n 정수 1~16 IDXn Counter 설정

x \* , 수치 , Label , \*±수치 , Label±수치 Jump Address (0 ~ 4095)

[Description]

Sequence Control Main Loop 명령이다.

IDX1 ~ IDX16 Loop Counter에 대해 Direct Value를 지정하여 실행 한다.

설정한 횟수 만큼 반복 실행후 Loop End Signal을 받아 다음 명령을 수행한다.

**[Excise]**

|  |
| --- |
| IDX1=5  IDX2=6  START  NOP -- PC=0  A: NOP -- PC=1  B: NOP -- PC=2  NOP -- PC=3  **JNI1 A** -- PC=4 자신을 실행하고 5+1회째 까지 Loop 반복 하고  5+2회째 다음 명령 수행  NOP -- PC=5  **JNI2 B** -- PC=6 자신을 실행하고 6+1회째 까지 Loop 반복 하고  6+2회째 다음 명령 수행  NOP -- PC=7  STPS -- PC=8  END |

### JNCn

[Function]

작성된 Sequence를 수행 후 참조 Register에 의해 Jump 또는 다음의 Sequence로 진행한다.

[Format]

JNCn x

n 정수 1 ~ 16 CFLG Register 설정

x \* , 수치 , Label , \*±수치 , Label±수치 Jump Address (0 ~ 4095)

[Description]

Sequence Control Main Loop 명령이다.

각 명령에 대해 CLFG = 1이라면 Jump를 수행하고 0이면 다음 명령을 수행한다.

**[Excise]**

|  |
| --- |
| START  NOP -- PC=0  A: NOP -- PC=1  NOP -- PC=2  NOP -- PC=3  **JNC1 A** -- PC=4 CFLG = 1 일 때, A로 Jump한다.  그렇지 때는 다음의 명령을 실행한다.  NOP -- PC=5  NOP -- PC=6  STPS -- PC=7  END |

### JET

[Function]

기술된 Sequence를 수행후 Timer End Signal에 따라 Loop 또는 다음 Sequence로 진행한다.

[Format]

JET x

Timer End Signal 참조

x \* , 수치 , Label , \*±수치 , Label±수치 Jump Address (0 ~ 4095)

[Description]

Sequence Control Main Timer Jump 명령이다.

JET 명령시 참조할 Timer 값을 설정한다.

해당 명령에서 지정된 시간이 경과전까지는 해당 명령에서 Jump를 수행하고

지정된 시간 경과후에는 다음 명령을 수행한다.

**[Excise]**

|  |
| --- |
| REGISTER  :  TIMER1 = 10MS  :  MODULE BEGIN  START #0  NOP T1  ST0: NOP  :  JET ST0  NOP  STPS  END |

### JET2

[Function]

기술된 Sequence를 수행후 Timer End Signal에 따라 Loop 또는 다음 Sequence로 진행한다.

[Format]

JET2 x

x \* , 수치 , Label , \*±수치 , Label±수치 Jump Address (0 ~ 4095)

[Description]

Sequence Control Main Timer Jump 명령이다.

T2명령시 참조할 Timer 값을 설정한다.

해당 명령에서 지정된 시간이 경과전까지는 해당 명령에서 Jump를 수행하고

지정된 시간 경과후에는 다음 명령을 수행한다.

**[Excise]**

|  |
| --- |
| TIMER2=1ms -- TMREF 時間設定  START  NOP T2 -- PC=0  A: NOP -- PC=1  NOP -- PC=2  **JET2 A** -- PC=3 이 줄을 실행 한 후 TIMER2에 설정된 시간  Loop 반복 설정 시간 초과 후 다음 명령 실행  NOP -- PC=4  NOP -- PC=5  STPS -- PC=6  END |

### JZD

[Function]

기술된 Sequence를 수행후 Jump 또는 다음 Sequence로 진행한다.

[Format]

JZD x

x \* , 수치 , Label , \*±수치 , Label±수치 Jump Address (0 ~ 4095)

[Description]

Sequence Control Main 조건 Jump 명령이다.

1회째 Data Generator에 Data Invert 명령을 보내고 지정 Address로 Jump한다.

2회째 다음 Sequence로 진행 한다.

**[Excise]**

|  |
| --- |
| START  NOP -- PC=0  A: NOP -- PC=1 JZD명령에 대한 목적 지점  NOP -- PC=2  NOP -- PC=3  JZD **A** -- PC=4 1회째 자신을 실행하고 Data Invert 명령을 보내고 A Label로  Jump한다. 2회째 자신을 실행하고 다음 명령을 수행한다.  STPS -- PC=5  END |

### JEXFLG

[Function]

기술된 Sequence를 수행후 참조 Register에 따라 Jump 또는 다음 Sequence로 진행한다.

[Format]

JEXFLG x

x \* , 수치 , Label , \*±수치 , Label±수치 Jump Address (0 ~ 4095)

[Description]

Sequence Control Main 조건 Jump 명령이다.

해당 명령을 수행할수 있도록 참조 Flag를 설정 한다.

EXINT,EXINT1 ~ JEXFLG7 명령과 함께 사용하며 해당 명령전에 EXINTn을 설정 한다.

각 명령에 대해 참조할 Bit가 1이면 Jump를 수행하고 0이면 다음 명령을 수행한다.

**[Excise]**

|  |
| --- |
| START  A: NOP -- PC=0 JEXFLG 명령에 대한 목적 지점  NOP EXINT7 -- PC=1  **JEXFLG A** -- PC=2 자신을 실행하고 EXINT7 의 값이 1xxxxxxxB 일때  A Label로 Jump 하고값이 0xxxxxxxB 일때  다음 명령 수행  NOP -- PC=3  NOP -- PC=4  STPS -- PC=5  END |

### JMFLG

[Function]

기술된 Sequence를 수행후 참조 Register에 따라 Jump 또는 다음 Sequence로 진행한다.

[Format]

JMFLG x

x \* , 수치 , Label , \*±수치 , Label±수치 Jump Address (0 ~ 4095)

[Description]

Sequence Control Main 조건 Jump 명령이다.

JMFLG의 명령을 수행할수 있도록 참조 Flag를 설정 한다.)

각 명령에 대해 참조할 Bit가 1이면 Jump를 수행하고 0이면 다음 명령을 수행한다.

**[Excise]**

|  |
| --- |
| START  NOP -- PC=0  NOP -- PC=1  **JMFLG A** -- PC=2 자신을 실행하고 CFLG 의 값이 x1xxB 일때  A Label로 Jump하고 값이 x0xxB 일때 다음 명령 수행  NOP -- PC=3  A: NOP -- PC=4 JMFLG3 명령에 대한 목적 지점  STPS -- PC=5  END |

### JLML

[Function]

기술된 Sequence를 수행후 Loop End Signal에 따라 Loop 또는 다음 Sequence로 진행한다.

[Format]

JLML x

x \* , 수치 , Label , \*±수치 , Label±수치 Jump Address (0 ~ 4095)

[Description]

Sequence Control Main Loop 명령이다.

**[Excise]**

|  |
| --- |
| START  NOP -- PC=0  A:NOP -- PC=1  NOP -- PC=2  NOP -- PC=3  **JLML A** -- PC=4 자신을 실행하고 . 조건 일치 JUMMP한다.NOP -- PC=5  NOP -- PC=6  STPS -- PC=7  END |

### JLMN

[Function]

기술된 Sequence를 수행후 Loop End Signal에 따라 Loop 또는 다음 Sequence로 진행한다.

[Format]

JLMN m x

m 정수 0 ~ $FFFFFFFF Match조건

x \* , 수치 , Label , \*±수치 , Label±수치 Jump Address (0 ~ 4095)

[Description]

Sequence Control Main Loop 명령이다.

**[Excise]**

|  |
| --- |
| START  NOP -- PC=0  A:NOP -- PC=1  NOP -- PC=2  NOP -- PC=3  **JLMN #AA A** -- PC=4 자신을 실행하고 . 조건 일치 JUMMP한다.NOP -- PC=5  NOP -- PC=6  STPS -- PC=7  END |
|  |

### VSBMPLEV

[Function]

기술된 Sequence에 VSBMP Level을 Setting한다.

[Format]

VSBMPLEV

[Description]

Sequence Control Main Bump 명령이다.

임의 지점에서 DPS Bump Level을 Setting하고 Program Counter의 값을 1 증가 시킨다.

**Rate 최소 시간은 10ns**이며 Data부에서 1 ~ 8의 Bump 전압을 미리 Setting해두고

이 명령에서 Setting된 전압을 출력할수있도록 Setting한다.

Power 출력 명령인 VSBMP 명령을 수행하기 전까지 **Default 850ns ~ 1us**가

필요하므로 주의 해야 한다.

**[Excise]**

|  |
| --- |
| START  NOP -- PC=0  NOP　　　 -- PC=1  VSBMPLEV=3 -- PC=2    VSBMP -- PC=3  NOP -- PC=5  STPS -- PC=6  END |

### VSBMP

[Function]

기술된 Sequence에 VSBMP Power On을 실행한다.

[Format]

VSBMP

[Description]

Sequence Control Main Bump 명령이다.

임의 지점에서 DPS Bump Power On을 실행하고 Program Counter의 값을 1 증가 시킨다.

**Rate 최소 시간은 10ns**이며 실행 전에 VSBMP Level이 설정 되어야 하고

설정이 되어 있지 않을시 초기 전압이 출력된다.

**[Excise]**

|  |
| --- |
| START  NOP -- PC=0  NOP　　　 -- PC=1  VSBMPLEV=3 -- PC=2    VSBMP -- PC=3  NOP -- PC=5  STPS |

### IOBMPLEV

[Function]

기술된 Sequence에 IOBMP Level을 Setting한다.

[Format]

IOBMPLEVn

n 정수 1 ~ 8 IO Bump Level 설정

[Description]

Sequence Control Main Bump 명령이다.

임의 지점에서 IO Bump Level을 Setting하고 Program Counter의 값을 1 증가 시킨다.

**Rate 최소 시간은 10ns**이며 Data부에서 1 ~ 8의 Bump 전압을 미리 Setting해두고

이 명령에서 Setting된 전압을 출력할수있도록 Setting한다.

Power 출력 명령인 IOBMP 명령을 수행하기 전까지 **Default 850ns ~ 1us**가

필요하므로 주의 해야 한다.

**[Excise]**

|  |
| --- |
| START  NOP TIMD2=10ms  NOP TIMER1=TIMD2 TMRST  IOBMPLEV=3    IOBMP -- PC=3 Rate 最小時間 10ns  JET .  NOP  STPS -- PC=6  END |

### IOBMP

[Function]

기술된 Sequence에 IOBMP Power On을 실행한다.

[Format]

VSBMP

[Description]

Sequence Control Main Bump 명령이다.

임의 지점에서 IO Bump Power On을 실행하고 Program Counter의 값을 1 증가 시킨다.

**Rate 최소 시간은 10ns**이며 실행 전에 IOBMP Level이 설정 되어야 하고

설정이 되어 있지 않을시 초기 전압이 출력된다.

수행후 Function Test는 충분히 Power가 안정된 후에 실행하기 위해 JETn을 이용하여

적정한 시간을 갖는것이 좋다.

**[Excise]**

|  |
| --- |
| START  NOO TIMD2=10ms  NOP TIMER1=TIMD2 TMRS  IOBMPLEV=3  IOBMP -- PC=3 Rate 最小時間 10ns  JET .  NOP -- PC=5  STPS -- PC=6  END |

## Sub Command Description

### IDXIm Set

[Function]

기술된 Sequence에서 IDXIm Register의 Setting을 한다.

[Format]

IDXm=n

m 정수 1 ~ 16 Loop값 저장 Register Number

n 정수 2 ~ $FFFFFFFF Loop 설정 값

[Description]

Sequence Control Sub 명령이다.

Loop Counter Register 1 ~ 16에 대해 Test 도중 설정 값을 변경 할수 있다.

IDXm 과 동일한 명령이며 도중에 변경된 값은 Test 종료 후에도 유지 된다.

실행시키고자 하는 Loop와 같은 Sequence에 Setting은 불가능 하다.

**[Excise]**

|  |
| --- |
| IDX3=$5 최초 IDX3 Register에 설정된 값은 5회  START  NOP -- PC=0  NOP -- PC=1  NOP IDX**I3=$9** -- PC=2 LDI3 Register설정된 값을 9회로 변경  JNI3 \* -- PC=3 Loop 9회 실행 하고  Loop 10회째 다음 명령 수행  IDX3=$9 동시 수행 불가  NOP -- PC=4  STPS -- PC=5  END |

### TIMDn Set

[Function]

기술된 Sequence에서 TIMDm Register의 Setting을 한다.

[Format]

TIMDm = n

m 정수 1 ~ 4 Timer값 저장 Register Number

n 정수 100ns ~ 409.6s (1Rate ~ 4096Rate) Timer 설정 값

[Description]

Sequence Control Sub 명령이다.

TIMDn Register 1 ~ 4에 대해 Test 도중 설정 값을 변경 할수 있다.

.TIMDn 과 동일한 명령이며 도중에 변경된 값은 Test 종료 후에도 유지 된다.

실행시키고자 하는 REFST,PAUST,TRAST,TMRST와 같은 Sequence에 Setting은 불가능 하다.

**[Excise]**

|  |
| --- |
| TIMD2=2ms 최초 TIMD2 Register에 설정된 시간은 2ms  STAR>  NOP **TIMD2=10ms** -- PC=0 TIMD2 Register에 설정된 시간은 10ms  NOP TMTMR=TIMD2 TMRST -- PC=1 Normal Timer Start  TMRTM=TIMD2 동시 수행 불가  NOP -- PC=2  JET TMTMR \* -- PC=3 10ms동안 Loop 반복  10ms 초과후 다음 명령 수행  NOP -- PC=4  NOP -- PC=5  STPS -- PC=6  END |

### T3

[Function]

작성된 Sequence에서 TIMER2 (Auto Refresh Timer)를 시작한다.

[Format]

T3

[Description]

Sequence Control Sub 명령이다.

TIMER2 (Auto Refresh Timer)를 기술 한 Sequence를 시작한다.  
MODE REFRESH 설정이 필요합니다. (Auto refresh mode)

T2와 함께 사용할 수 없다.

**[Excise]**

### T2

[Function]

작성된 Sequence에서 TIMER2를 시작한다.

[Format]

T2

[Description]

Sequence Control Sub 명령이다.

TIMER2 (T2)를 기술 한 Sequence를 시작한다.  
  
T3와 함께 사용할 수 없다.

**[Excise]**

### T1

[Function]

작성된 Sequence에서 TIMER1을 시작한다.

[Format]

T1

[Description]

Sequence Control Sub 명령이다.  
  
T1을 묘사 한 Sequence에 시작한다.

**[Excise]**

|  |
| --- |
| TIMER1=6ms  TIMER2=10ms  START  A: NOP T1  NOP  JET A  B: NOP T2  JET2 B  NOP  STPS  END |

### INTDS

[Function]

기술된 Sequence에 Interrrupt Disable을 한다.

[Format]

INTDS

[Description]

Sequence Control Sub 명령이다.

임의 지점에서 Refresh Interrupt에 의한 Interrupt Routine으로 Jump하는것을 임시 막아준다.

**[Excise]**

### EXINT

[Function]

기술된 Sequence를 수행후 참조 Register에 따라 Jump 또는 다음 Sequence로 진행한다.

[Format]

EXINT

[Description]

Sequence Control Sub 명령이다.

**[Excise]**

# ADDRESS PATTERN 발생 명령

* + **ADDRESS 발생부의 Register**

Address Register는 Address 발생에 관련된 Register들입니다.

각 Register의 설정과 관련해서는 아래 표 Address Control Register List를 참고 하십시오.

★표 6-1 Address Register List

|  |  |  |  |
| --- | --- | --- | --- |
| Register | Description | Bits | Set |
| XB, YB, Z | 각 Address의 Base Register입니다. 주로 Test 대상 Cell의 Address 발생에 사용됩니다. | 24 | R/  W |
| XH, YH, ZH | 각 Address의 Base Register에 Load하는 초기치를 Store하는 Register입니다. | 24 | R/  W |
| XC,YC | Test 대상 Cell과의 상호 관계를 Test하기 위한  Address(Disturb Address)의 발생 및 Base Register에 의해서 발생할 수 없는 Test Pattern Address의 발생에 사용됩니다. |  |  |
| XS, XK,  YS, YK | 주로 XC, YC 의 각 Address를 일시적으로 퇴피하기 위한 Save Register로서 사용됩니다.  XC, YC Register와 동등의 기능을 가지고 있으므로, XC, YC와 조합해 사용하는 것으로, 보다 복잡한 Address 발생이 가능해집니다. | 24 | R/  W |
| D1A(D1),D2A(D2),  D1B~ D1H,  D2B~ D2D | XB, YB, XC, YC, XS, YS, XK, YK의 Address 연산을 실시할 때의 Offset Data로서 사용합니다. | 24 | R/  W |
| D3, D4 | XC, YC, XS, YS, XK, YK의 Address 연산을 실시할 때의 Offset Data로서 사용합니다. | 24 | R/  W |
| D3B, D4B | D3, D4Register에 Load하는 초기치를 Store하는 Register입니다. | Time | R/  W |
| LMAX, HMAX | LMAX, HMAX는, 조건부 연산을 사용할 때의 X, Y  Register의 유효 Bit를 지정하는 Register 입니다. | Time | R/  W |
| N, NH, ND1 ~ ND7 | 주로 Bank Address의 발생에 사용합니다.  NH는 N Register에 Load하는 초기치를 Store하는 Register입니다.  ND1 ~ ND7는 N Register의 연산에 사용하는 Register입니다. | Time | R/  W |
| B, BH, BD1 ~ BD7 | 주로 Bank Address의 발생에 사용합니다.  BH는 B Register에 Load하는 초기치를 Store하는 Register입니다.  BD1 ~ BD7는 B Register의 연산에 사용하는 Register입니다. | Time | R/  W |
| ZD | Z Address의 연산에 사용하는 Register입니다.  Z Address를 DUT에 인가할 수 없습니다만, ALPG 내부에서 Z Address를 발생해, X/Y Address에 끼어들게 하는 것이나, Burst Address의 발생에 사용할 수 있습니다. | Time | R/  W |
| XMAX, YMAX, ZMAX | X0-17, Y0-17, Z0-17에 출력하는 각 Address의 유효 Bit를 나타내는 Register입니다. | Time | R/  W |
| XOS, YOS | Address에 대해서 가산하는 Offset치를 Store하는 Register입니다. Address(0, 0)를 원점으로 한 작은 영역에 대해서 Program 한 Pattern을, 다른 영역으로 이동할 때 사용합니다. | Time | R/  W |
| XT, YT | Address Line에 대해서 일시적으로 논리 Data를 출력할 때에, 그 논리 Data를 Store하는 Regsiter입니다. Address Line으로부터 mode 설정을 실시하는 디바이스를 측정할 때 등에 사용합니다. | Time | R/  W |

**[Memo]**

► “ “ 표시가 있는 Register는 Test Program REGISTER문과 Test Plan program의 REG문에서 값을 설정할 수 있습니다.

► “ “표시가 있는 Register는 Pattern Program의 DSET 명령, Test Program Register문과 Test Plan Program의 REG문에서 값을 설정할 수 있습니다.

* + **Address 발생 명령의 구성**

**[ 개요 ]**

Address 발생 부는, 다음의 각 명령으로 구성되어 있습니다.

► Base address field

► Current address field

► Source address field

► N register field

► B register field

► Address inversion field

► Refresh counter field

► D3, D4 register field

► Address swap field

**[ Description ]**

Address 발생 명령부의 각 명령의 mnemonic을 아래에 기술 하였습니다.

괄호로 묶여진 Group내에서 동일 Cycle에는 1개의 명령선택이 가능 합니다.

사각형은 1개의 명령선택을 의미 합니다.

**[ Mnemonic ]**

**▶Base Address Field**

* X base address field aX
* Y base address field aY

XB<XB

XB<XB+1

XB<XB-1

XB<XH

XB<0

XB<aX

XB<XB+aX

XB<XB-aX

D1A(D1)

D1B

D1C

D1D

D1E

D1F

D1G

D1H

D2A (D2)

D2B

D2C

D2D

YB<YB

YB<YB+1

YB<YB-1

YB<YH

YB<YB+1^BX

YB<YB-1^BX

YB<YB+1+BX

YB<YB-1-BX

YB<0

YB<aY

YB<YB+aY

YB<YB-aY

YB<YB+aY^BX

YB<YB-aY^BX

YB<YB+aY+BX

YB<YB-aY-BX

D1A(D1)

D1B

D1C

D1D

D1E

D1F

D1G

D1H

D2A(D2)

D2B

D2C

D2D

* Z base address field

Z<Z

Z<Z+1

Z<Z-1

Z<ZH

Z<Z\*2

Z</Z

Z<0

Z<Z+1^BX

Z<Z-1^BX

Z<Z+1^CY

Z<Z-1^CY

Z<Z+ZD

Z<Z-ZD

Z<Z+ZD^BX

Z<Z-ZD^BX

Z<Z+ZD^CY

Z<Z-ZD^CYZ<Z-ZD^CY \*1

**[Memo]**

► D1A는 D1로 기술할 수 있습니다.

► D2A는 D2로 기술할 수 있습니다.

► D1A ~ D1H Register는, 동일 Step에 2 종류 이상 기술할 수 없습니다.

► D2A ~ D2D Register는, 동일 Step에 2 종류 이상 기술할 수 없습니다.

**▶Current Address Field**

* X current address field

fX aX bX

XC<XC

XC< fX

XC<XS

XC<XK

XC<XC

XC<XC

XC<XC

XC<XC

XCS<fX\*1

XCK< fX\*1

XCSK< fX\*1

XC<XS< fX

XS<XC< fX

XC<XK< fX

XK<XC< fX

XC<>XS

XC<>XK

XS<XS

XS<XS

XS<XS

XS<XS

XS< fX

XS<XC

XS<XS

XS<XS

XS<XS

XS<XS

XS<XS

XS<XS

XK<XK

XK<XK

XK<XK

XK<XK

XK<XK

XK<XK

XK< fX

XK<XC

XK<XK

XK<XK

XK<XK

XK<XK

aX

aX+1

aX-1

aX\*2

/aX

bX

aX+bX

aX-bX

aX&bX

aX"bX

aX'bX

0

XB

XC

XS

XK

D1A(D1)

D1B

D1C

D1D

D1E

D1F

D1G

D1H

D2A(D2)

D2B

D2C

D2D

D3

D4

* Y current address field

fY aY bY

YC<YC

YC< fX

YC<YS

YC<YK

YC<YC

YC<YC

YC<YC

YC<YC

YCS< fY \*1

YCK< fY \*1

YCSK< fY \*1

YC<YS< fY

YS<YC< fY

YC<YK< fY

YK<YC< fY

YC<>YS

YC<>YK

YS<YS

YS<YS

YS<YS

YS<YS

YS< fY

YS<YC

YS<YS

YS<YS

YS<YS

YS<YS

YS<YS

YS<YS

YK<YK

YK<YK

YK<YK

YK<YK

YK<YK

YK<YK

YK< fY

YK<YC

YK<YK

YK<YK

YK<YK

YK<YK

aY

aY+1

aY-1

aY\*2

/ aY

bY

aY+bY

aY-bY

aY&bY

aY"bY

aY' bY

0

aY +1+CY

aY -1-CY

aY+ bY+CY

aY- bY-CY

aY+1+BX

aY-1-BX

aY+ bY+BX

aY- bY-BX

aY+1^CY

aY-1^CY

aY+ bY^CY

aY- bY^CY

aY+1^BX

aY-1^BX

aY+ bY^BX

aY- bY^BX

YB

YC

YS

YK

D1A(D1)

D1B

D1C

D1D

D1E

D1F

D1G

D1H

D2A(D2)

D2B

D2C

D2D

D3

D4

**[ Description ]**

\*1 XC, XS, XK의 각 Register에, F의 연산 결과를 동시에 Load하는 명령입니다.

1. XCS<fX ; XC, XS Register에 Load

2. XCK<fX ;XC, XK Register에 Load

3. XCSK<fX ;XC, XS, XK Register에 Load

YC, YS, YK에 대해서도 같습니다.

XCS<fX fX: aX+bX aX: XB bX: D1를 선택했을 때

↓

XCS<XB+D1 ; Programming시의 Mnemonic

**[Memo]**

► D1A는 D1와 기술할 수 있습니다.

► D2A는 D2와 기술할 수 있습니다.

► D1A ~ D1H Register는, 동일 Step에 2 종류 이상 기술할 수 없습니다.

► D2A ~ D2D Register는, 동일 Step에 2 종류 이상 기술할 수 없습니다.

▶Source Address Field

X<XB

X<XC

X<XS

X<XK

X<YB

X<YC

X<YS

X<YK

X<XB'Z

X<XC'Z

X<XS'Z

X<XK'Z

X<Z

Y<YB

Y<YC

Y<YS

Y<YK

Y<XB

Y<XC

Y<XS

Y<XK

Y<XB'Z

Y<XC'Z

Y<XS'Z

Y<XK'Z

Y<Z

**[Memo]**

X address 발생의 Register(XB, XC, XS, XK)의 그룹1, Y address 발생의 Register의(YB, YC, YS, YK)의 그룹2에서, 같은 그룹 내에서 다른 Register를 선택 할 수 없습니다.

X<XB Y<XK(불가)

X<YB Y<YK(불가)

X<XC Y<XC(가능)

X<XC Y<YC(가능)

▶N/B Register Field

**N Register Field B Register Field**

N<N

N<N+1

N<N-1

N<NH

N<0

N</N

N<NDn\*1

N<N+ NDn \*1

N<N- NDn\*1

B<B

B<B+1

B<B-1

B<BH

B<0

B</B

B< BDn\*1

B<B+ BDn\*1

B<B- BDn\*1

**[Memo]**

\*1 n은 1부터 7까지 입니다.

ND1는 ND와 기술할 수 있습니다.

BD1는 BD와 기술할 수 있습니다.

▶Other Field

**Address inversion field**

[/X] [/Y] [/Z]

**D3,D4 register field**

**Address swap field**

[LMAX<>HMAX ]

D3<D3

D3<D3\*2

D3<D3B

D3<D3\*2L

D3<D3+1

D3<D3-1

D4<D4

D4<D4\*2

D4<D4/2

D4<D4B

D4<D4\*2L

* + **Base Address Field**

**[Function]**

Base address field는, N2, N3/2계의 Test Pattern에 대해 Test 대상 Cell의 Address를 발생하기 위한 연산 명령입니다

**[Format ]**

**Hold : XB<XB, YB<YB, Z<Z**

**Clear : XB<0, YB<0, Z<0**

**Load : XB<XH, YB<YH, Z<ZH , XB<Dn , YB<Dn**

**INC : XB<XB+1, YB<YB+1, Z<Z+1**

**DEC : XB<XB-1, YB<YB-1, Z<Z-1**

**가감산명령 : XB<XB±Dn, YB<YB±Dn, Z<Z±ZD**

**조건부 연산 명령 참조(표 6.1)**

**Shift : Z<Z\*2**

**Complement : Z</Z**

**[Description ]**

**[Exercise ]**

**▶Hold (XB<XB YB<YB Z<Z)**

**[Function]**

Base Address 의 값을 hold하는 기능을 합니다.

**[Format ]**

**XB<XB**

XB 에 XB의 값을 set함으로 값의 변화가 없습니다.

**YB<YB**

YB 에 YB의 값을 set함으로 값의 변화가 없습니다.

**Z<Z**

Z 에 Z의 값을 set함으로 값의 변화가 없습니다.

**[Description ]**

아무것도 실행하지 않는 경우에 사용합니다.

이 명령은 생략 할 수 있습니다.

**[Exercise ]**

NOP XB<0 YB<0

LB0: JNI1 LB0 W XB<XB+1 YB<YB+1^BX X<XB Y<YB

|  |
| --- |
| NOP XC<XB+1 YC<YB+1^BX X<XB Y<YB |

|| 같다. (기술 생략이 가능하다.)

|  |
| --- |
| NOP XB<XB YB<YB XC<XB+1 YC<YB+1^BX X<XB Y<XB |

**▶Clear (XB<0 YB<0 Z<0)**

**[Function]**

XB, YB, Z의 Initialize에 사용합니다.

**[Format ]**

**XB<0**

XB에 0을 Set합니다.

**YB<0**

YB에 0을 Set합니다.

**Z<0**

Z에 0을 Set합니다.

**[Description ]**

**[Exercise ]**

MPAT ~~CHECK~~

REGISTER

XMAX=#7F

YMAX=#7F

IDXI=#3FFF

PC=#0

START #00

|  |  |
| --- | --- |
| NOP XB<0 YB<0 |  |

LB0: JNI1 LB0 W XB<XB+1 YB<BY+1^XB X<XB Y<YB

XB, YB, Z를 사용하는 경우는

1STEP전에 INITIAL

**▶Load (XB<XH YB<YH Z<ZH XB<Dn YB<Dn)**

**[Function]**

XB, YB, Z의 Initialize에 사용합니다.

본 명령에 의해서 XH, YH, ZH, Dn의 각 Register의 내용을 각각 XB, YB, Z에 Load합니다.

**[Format ]**

**XB<XH**

XB에 XH Register의 값을 Set한다.

**YB<YH**

YB에 YH Register의 값을 Set한다.

**Z<ZH**

Z에 ZH Register의 값을 Set한다.

**XB<Dn**

XB에 Dn Register의 값을 Set한다.

**YB<Dn**

YB에 Dn Register의 값을 Set한다.

Dn는, D1A ~D1H, D2A~D2D를 나타냅니다.

**[Description ]**

**[Exercise ]**

**▶INC (XB<XB+1 YB<YB+1 Z<Z+1)**

**[Function]**

XB, YB, Z의 각 address의 Increment에 사용합니다

**[Format ]**

**XB<XB+1**

XB에 XB+1의 값을 Set합니다..

**XB<XB+1**

YB에 YB+1의 값을 Set 합니다.

**Z<Z+1**

Z에 Z+1의 값을 Set합니다.

XB, YB, Z Register를 X, Y, Z Address에 출력하는 경우, XMAX, YMAX, ZMAX로 지정된 유효 Bit의 범위에서 출력됩니다.

**[Description ]**

**[Exercise ]**

본 명령에 의해서 다음의 Address 순서를 발생할 수 있습니다.

★그림6-1

► LB0: JNIn LB0 XB<XB+1 X<XB

0 X XMAX

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

► LB0: JNIn LB0 YB<YB+1 Y<YB

► LB0: JNIn LB0 XB<XB+1 YB<YB+1 X<XB Y<YB

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

0

Y

YMAX

0 X XMAX

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

► LB0: JNIn LB0 XB<XB+1 YB<YB+1 X<XB Y<YB

JNIm LB0 XB<XB+1 YB<YB+1 Z<Z+1 X<XB Y<YB

0 X XMAX 0 X XMAX

0

X

YMAX

0

Y

YMAX

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

Z=0

Z=1

Z=2

Z=3

**▶DEC (XB<XB-1 YB<YB-1 Z<Z-1)**

**[Function]**

XB, YB, Z의 각 Address의 Decrement에 사용합니다.

**[Format ]**

**XB<XB-1** (XB)-1 -> XB

**YB<YB-1** (YB)-1-> YB

**Z<Z-1**  (Z)-1 -> Z

**[Description ]**

**[Exercise ]**

★그림6-2

► LB0: JNIn LB0 XB<XB-1 X<XB

0 X XMAX

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

► LB0: JNIn LB0 YB<YB-1 Y<YB

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

0

Y

YMAX

► LB0: JLN1 #1F LB0 XB<XB-1 YB<YB-1 X<XB Y<YB

0 X XMAX

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

► LB0: JNIn LB0 XB<XB-1 YB<YB-1 X<XB Y<YB

JNIn LB0 XB<XB-1 YB<YB-1 Z<Z-1 X<XB Y<YB

0 X XMAX 0 X XMAX

0

X

YMAX

0

Y

YMAX

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

Z=0

Z=1

Z=2

Z=3

**▶가감산명령(XB<XB±Dn YB<YB±Dn Z<Z±ZD)**

**[Function]**

XB, YB, Z의 각 Address의 가산 및 감산에 사용합니다.

**[Format ]**

**XB<XB±Dn** XB±Dn → XB

**YB<YB±Dn** YB±Dn → YB

**Z<Z±ZD** Z±ZD → Z

Dn는, D1A ~ D1H, D2A ~ D2D를 나타냅니다.

**[Memo]**

XB, YB, Z Register를 X, Y, Z Address에 출력하는 경우, XMAX, YMAX, ZMAX로 지정한 유효 Bit의 범위에서 출력됩니다.

**▶조건부 연산 명령**

**[Function]**

이 명령은, X, Y, Z의 각 Address를 연결해 Address를 발생하는 경우에 사용합니다.

**[Format]**

* 표 6.1 조건부 연산

|  |  |  |  |
| --- | --- | --- | --- |
| 조건부 발생 명 | 조건 | 판정 | 연산 |
| YB<YB+1^BX | XB = LMAX에 따르면 CARRY / BORROW 발생 | then | YB+1->YB(XB←0) |
| else | YB->YB |
| YB<YB-1^BX | XB = LMAX에 따르면 CARRY / BORROW 발생 | then | YB-1->YB(XB←0) |
| else | YB->YB |
| YB<YB+1+BX | XB = LMAX에 따르면 CARRY / BORROW 발생 | then | YB+2->YB(XB←0) |
| else | YB+1->YB |
| YB<YB-1-BX | XB = LMAX에 따르면 CARRY / BORROW 발생 | then | YB-2->YB(XB←0) |
| else | YB-1->YB |
| Z<Z+1^BX | XB = LMAX and YB = HMAX에 따르면 CARRY / BORROW 발생 | then | Z+1->Z(XB←0,YB←0) |
| else | Z->Z |
| Z<Z-1^BX | XB = LMAX and YB = HMAX에 따르면 CARRY / BORROW 발생 | then | Z-1->Z(XB←0,YB←0) |
| else | Z->Z |
| Z<Z+1^CY | XC (S) (K) = LMAX and YC (S) (K) = HMAX에 따르면 CARRY / BORROW 발생 | then | Z+1->Z(XC←0,YC←0) |
| else | Z->Z |
| Z<Z-1^CY | XC (S) (K) = LMAX and YC (S) (K) = HMAX에 따르면 CARRY / BORROW 발생 | then | Z-1->Z(XC←0,YC←0) |
| else | Z->Z |
| YB<YB+Dm^BX | XB = LMAX에 따르면 CARRY / BORROW 발생 | then | YB+Dn->YB(XB←0) |
| else | YB->YB |
| YB<YB-Dn^BX | XB = LMAX에 따르면 CARRY / BORROW 발생 | then | YB-Dn->YB(XB←0) |
| else | YB->YB |
| YB<YB+D<+BX | XB = LMAX에 따르면 CARRY / BORROW 발생 | then | YB+Dn+1->YB(XB←0) |
| else | YB->YB |
| YB<YB-Dn-BX | XB = LMAX에 따르면 CARRY / BORROW 발생 | then | YB-Dn-1->YB(XB←0) |
| else | YB-Dn->YB |
| Z<Z+ZD^BX | XB = LMAX and YB = HMAX에 따르면 CARRY / BORROW 발생 | then | Z+ZD->Z(XB←0,YB←0) |
| else | Z->Z |
| Z<Z-ZD^BX | XB and YB = HMAX에 따르면 CARRY / BORROW 발생 | then | Z-ZD->Z(XB←0,YB←0) |
| else | Z->Z |
| Z<Z+ZD^CY | XC (S) (K) = LMAX and YC (S) (K) = HMAX에 따르면 CARRY / BORROW 발생 | then | Z+ZD->Z(XC←0,YC←0) |
| else | Z->Z |
| Z<Z-ZD^CY | XC (S) (K) = LMAX and YC (S) (K) = HMAX에 따르면 CARRY / BORROW 발생 | then | Z-ZD->Z(XC←0,YC←0) |
| else | Z->Z |

Dn는, D1A ~ D1H, D2A ~ D2D를 나타냅니다.

**[Description]**

Carry/Borrow-는, 이하의 Register 연산에 의한 Carry/Borrow-입니다.

► XB, YB연산에 의한 Carry/Borrow

조건부 연산 명령과 동일 Cycle에 기술된 XB, YB Register의 연산에 대하 고, LMAX, HMAX에 Store한 유효 Bit의 최상위Bit에 의해 발생하는 Carry / Borrow.

► XC(XS)(XK), YC(YS)(YK) 연산에 의한 Carry/Borrow-

조건부 연산 명령과 동일 Cycle에 기술된 XC(XS)(XK), YC(YS)(YK) Register의 연산에 있고, LMAX, HMAX 에 Store한 유효 Bit의 최상위Bit에 의해 발생하는 Carry/Borrow.

본 명령에 의해서 X, Y, Z의 연결한 Address 발생을 실시할 수 있습니다. 일례로서

본 명령을 이용해 Address 순서의 발생을 나타냅니다.

**[Exercise]**

★그림 6-3

► LB0: JNIn LB0 XB<XB+1 YB<YB+1^BX X<XB Y<YB

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

0 X LMAX

0

Y

HMAX

► LB0: JNIn LB0 XB<XB-1 YB<YB-1^BX X<XB Y<YB

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

0 X LMAX

0

Y

HMAX

* LB0: JNIn LB0 XB<XB+1 YB<YB+1^BX Z<Z+1^BX X<XB Y<YB

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

0 X LMAX 0 X LMAX

0

X

HMAX

0

Y

HMAX

Z=0

Z=1

Z=2

Z=3

► LB0: JNIn LB0 XB<XB+1 YB<YB+1+BX X<XB Y<YB

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

0 X LMAX

0

Y

HMAX

▶**Shift (Z<Z\*2)**

**[Function]**

이 명령은, Z Register의 내용을 2배(1Bit 왼쪽 Shift)로 합니다.

**[Format]**

**Z<Z\*2**

; Z\*2→Z Shift left (LSB=0)

**[Description]**

**[Exercise]**

**▶Complement (Z</Z)**

**[Function]**

이 명령은, Z Register의 내용을 반전시킵니다.

**[Format]**

**Z</Z**

; NOT. (Z)→Z

**[Description]**

**[Exercise]**

* + **Current Address Field**

**[Function]**

Current address field는, N2, N3/2계의 Test Pattern에 대해 Test 대상 Cell과 상호 관계를 Test 하기 위한 Address(Disturb Address )의 발생 및 Base field의 연산에 의해 서 발생할 수 없는 N계 의 Test Pattern에 있어서의 Test 대상 Cell의 Address를 발생 하는 경우에 사용됩니다.

**[Format]**

3 종류의 명령 군으로 구성되어 있습니다.

► C[S][K]<f (B)

► C[S][K]<f (C[S][K])

► C[S][K]<f (B, C[S][K], t)

B는, XB, YB를 나타냅니다.

C[S][K]는 XC, YC, XS, YS 또는 XK, YK를 나타냅니다.

**[Description]**

Current Address Field의 설명으로 사용하는 Carry/Borrow-는, 이하의 Register 연산에 의한 Carry/Borrow-입니다.

► XB, YB연산에 의한 Carry/Borrow-

조건부 연산 명령과 동일 Cycle에 기술된 XB, YB Register의 연산에 대해, LMAX, HMAX에 Store한 유효 Bit의 최상위Bit에 의해 발생하는 Carry/Borrow-.

► XC(XS)(XK), YC(YS)(YK) 연산에 의한 Carry/Borrow-

조건부 연산 명령과 동일 Cycle에 기술된 XC(XS)(XK), YC(YS)(YK) Register의 연산에대해, LMAX, HMAX에 저장한 유효 Bit의 최상위Bit에 의해 발생하는 Carry/ Borrow-.

* + - **C[S][K]<f (B)**

**[Function]**

C[S][K]<f (B)는XC, YC또는XS, YS, XK, YK에 저장되는 Disturb Cell의 Address를 XB, YB에 저장되는 Test 대상 Cell의 Address로부터의 연산에 의해서 얻기 위한 명령 군 입니다.

**[Format]**

**C[S][K]< B**

**C[S][K]< B ± 1**

**C[S][K]< B ± Dn**

**C[S][K]< B \* 2** 🡨Shift left

**C[S][K]< / B** 🡨Complement

**C[S][K]< B & Dn** 🡨논리적

**C[S][K]< B " Dn** 🡨배타적 논리합

**C[S][K]< B ' Dn** 🡨논리합

**C[S][K]< B ± 1^BX**

**C[S][K]< B ± Dn^BX**

**C[S][K]< B ± 1 ± BX**

**C[S][K]< B ± Dn ± BX** 🡨YC또는 YS,YK만 유효

**C[S][K]< B ± 1^CY**

**C[S][K]< B ± Dn^CY**

**C[S][K]< B ± 1 ± CY**

**C[S][K]< B ± Dn ± CY**

▶**C[S][K]<B**

**[Function]**

XC(XS)(XK), YC(YS)(YK)의 각 Register의 Initialize에 사용됩니다. XB, YB의 각 Register의 내용이 XC(XS)(XK), YC(YS)(YK)의 각 Register에 Load 됩니다.

**[Format]**

**XC<XB YC<YB**

**XS<XB YS<YB**

**XK<XB YK<YB**

**[Description]**

**[Exercise]**

▶ **C[S][K]<B±1**

**[Function]**

XB, YB 에 저장된 Test 대상 Cell의 Address의 전후(±1)의 Address를 발생하는 경우에 사용해 주세요.

**[Format]**

**XC(XS)(XK)<XB±1,**

**YC(YS)(YK)<YB±1**

**[Description]**

XB±1, YB±1의 값이 XC(XS)(XK), YC(YS)(YK)에 Load 됩니다. 본 명령은, Test 대상 Cell의 주위만을 액세스 하는 Surround-disturb pattern의 발생, Galloping, Walking등의 N2계의 Test Pattern에 있어서의 Disturb Cell의 Address 발생의 Initialize에 사용해 주세요.

**[Exercise]**

★그림6-4

►**Surround Disturb**

NOP XC<XB YC<YB-1

NOP XC<XB+1 YC<YB-1

NOP XC<XB+1 YC<YB

NOP XC<XB+1 YC<YB+1

NOP XC<XB YC<YB+1

NOP XC<XB-1 YC<YB+1

NOP XC<XB-1 YC<YB

NOP XC<XB-1 YC<YB-1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | 8 | 1 | 2 |  |
|  | 7 |  | 3 |  |
|  | 6 | 5 | 4 |  |
|  |  |  |  |  |

►Galloping

R0 address 발생시, X address는 XC<XB+1를 사용

NOP XB<0 YB<0 TP<TPH

LB0: JNI1 LB0 W B<XB+1 YB<YB+1^BX X<XB Y<YB

LB1: NOP W XC<XB+1 YC<YB+1^BX X<XB Y<YB /D

LB 2: NOP R X<XC Y<YC

**:**

▶C[S][K]<B±Dn

**[Function]**

XB, YB에 저장된 Test 대상 Cell을 기준으로 다른 Cell의 Address를 발생시키는 경우에 사용됩니다.

**[Format]**

**XC<XB±Dn YC<YB±Dn**

**XS<XB±Dn YS<YB±Dn**

**XK<XB±Dn YK<YB±Dn**

(Dn=D1A ~ D1H, D2A ~ D2D, D3, D4)

**[Description]**

**[Exercise]**

명령은, D3 Register의 연산 기능과 조합해 Butterfly pattern의 발생 등에 사용됩니다. 이 아래에 Butterfly pattern의 예를 나타냅니다.

★그림 6-5 Butterfly pattern

위 번호 순서대로 Test 대상 Cell로부터 Disturb Cell로 Dn을 이용해 이동한다.

|  |
| --- |
| REGISTER  D3B=#1  :  START #00  NOP XB<0 YB<0 TP<TPH D3<D3B  LB0: JNI1 LB0 W XB<XB+1 YB<YB+1^BX X<XB Y<YB  LB1: NOP W YC<YB-D3 X<XB Y<YB /D  LB2: NOP R X<XB Y<YC  NOP R XC<XB+D3 X<XB Y<YB /D  NOP R X<XC Y<YB  NOP R YC<YB+D3 X<XB Y<YB /D  NOP R X<XB Y<YC  NOP R XC<XB-D3 X<XB Y<YB /D  NOP R X<XC Y<YB D3<D3+1  JNI2 LB2 R YC<YB-D3 X<XB Y<YB /D  JNI3 LB1 W XB<XB+1 YB<YB+1 X<XB Y<YB D3<D3B |

**▶** C[S][K]<B±1, Dn^BX

**[Function]**

XB와 YC, YS 또는 YK를 연결해서 사용하는 경우의 명령입니다.

**[Format]**

**YC< YB±1^BX YC<YB±Dn^BX**

**YS< YB±1^BX YS<YB±Dn^BX**

**YK< YB±1^BX YK<YB±Dn^BX**

(Dn=D1A ~ D1H, D2A ~ D2D, D3, D4)

**[Description]**

XB Carry가 발생하지 않는 경우 YB

XB Carry가 발생했을 경우 YB±1, Dn

본 명령은, Galloping, Walking, Galwrec등의 N2계의 Test Pattern의 발생을 XB와 YC 또는 YS를 연결해 실시하는 경우에 사용해 주세요.

Ex1.

★그림 6-6

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

0 X LMAX

0

Y

HMAX

YB→YC

YB+1→YC

**[Exercise]**

**▶** **C[S][K]<B±1, Dn±BX**

**[Function]**

YC, YS 및 YK의 연산 명령에 대해 XB의 연산으로 발생한 Carry 및 Borrow-를 가산, 감산 하는 명령입니다.

**[Format]**

**YC< YB±1±BX YC<YB±Dn±BX**

**YS< YB±1±BX YS<YB±Dn±BX**

**YK< YB±1±BX YK<YB±Dn±BX**

(Dn=D1A ~ D1H, D2A ~ D2D, D3, D4)

**[Description]**

XB Carry/Borrow-가 발생하지 않는 경우 YB±1, Dn

XB Carry/Borrow-가 발생했을 경우 YB±1, Dn±1

주로, Test 대상 Address로부터 아래와 같이 나타내는 Disturb 개시 Address를 설정 할 경우에 본 명령이 필요합니다.

Ex1.

★그림 6-7

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| ★ |  | ★ |  |
|  |  |  |  |
|  |  |  |  |

0 X LMAX

0

Y

HMAX

1 : XC<XB+1 YC<YB+1

2 : XC<XB+1 YC<YB+1+BX

**[Exercise]**

**▶** **C[S][K]<B±1, Dn^CY**

**[Function]**

YC, YS 및 YK의 조건부 연산 명령은, XC, XS 또는 XK의 연산에 대해 발생한 Carry에 의해서 YC와 YS, YK의 연산을 제어합니다.

**[Format]**

**YC< YB±1^CY YC<YB±Dn^CY**

**YS< YB±1^CY YS<YB±Dn^CY**

**YK< YB±1^CY YK<YB±Dn^CY**

(Dn=D1A ~ D1H, D2A ~ D2D, D3, D4)

**[Description]**

XC(XS)(XK) Carry가 발생하지 않는 경우 YB

XC(XS)(XK) Carry가 발생했을 경우 YB±1, Dn

본 명령도 C[S][K]<B±1^BX와 같게 Test 대상 Address로부터의 Disturb 개시 Address를 지정할 때 사용해 주세요.

C[S][K]<B±1, Dn^BX는 Test 대상 Address의 XB연산으로 Carry가 발생했을 때 YB±1, Dn를 실행합니다만, 본 명령은, Test 대상 Address의 XB연산으로 Carry가 발생하지 않을 때에도 XC, XS 또는 XK의 Carry가 발생했을 때에 YB±1, Dn를 실행합니다.

**[Exercise]**

**▶** **C[S][K]<B±1, Dn±CY**

**[Function]**

YC, YS 및 YK 의 연산 명령에, XC, XS 또는 XK 의 연산으로 발생한 Carry를 가산, 감산하는 명령입니다

**[Format]**

**YC< YB±1±CY YC<YB±Dn±CY**

**YS< YB±1±CY YS<YB±Dn±CY**

**YK< YB±1±CY YK<YB±Dn±CY**

(Dn=D1A ~ D1H, D2A ~ D2D, D3, D4)

**[Description]**

XC(XS)(XK) Carry/Borrow-가 발생하지 않는 경우 YB±1, Dn

XC(XS)(XK) Carry/Borrow-가 발생했을 경우 YB±1, Dn±1

본 명령도 C[S][K]<B±1±BX와 같게 Test 대상 Address로 부터의 Disturb 개시 Address를 지정했을 때에 사용해 주세요.

**[Exercise]**

**▶** **C[S][K]<B\*2**

**[Function]**

본 명령은, XB, YB 에 저장되어 있는 Address를 Shift left(XB\*2, YB\*2, LSB=0)하고 XC(XS)(XK), YC(YS)(YK)에 저장할 경우에 사용해 주세요.

**[Format]**

**XC< XB\*2 YC<YB\*2**

**XS< XB\*2 YS<YB\*2**

**XK< XB\*2 YK<YB\*2**

**[Description]**

**[Exercise]**

**▶** **C[S][K]</B**

**[Function]**

본 명령은, XB, YB에 저장되어 있는 Address를 반전한 Address를, XC(XS)(XK), YC(YS)(YK) 에 저장 할 경우에 사용해 주세요.

**[Format]**

**XC< /XB YC</YB**

**XS< /XB YS</YB**

**XK< /XB YK</YB**

**[Description]**

**[Exercise]**

**▶** **C[S][K]< B&Dn**

**[Function]**

본 명령은, XB 및 YB에 저장 되어 있는 Address와 Dn에 나타나는 Data의 논리(AND)를 취한 Address를 XC(XS)(XK) 및 YC(YS)(YK)에 저장할 경우에 사용해 주세요.

**[Format]**

**XC<XB&Dn YC<YB&Dn**

**XS<XB&Dn YS<YB&Dn**

**XK<XB&Dn YK<YB &Dn**

(Dn=D1A ~ D1H, D2A ~ D2D, D3, D4)

**[Description]**

**[Exercise]**

**▶** **C[S][K]< B'Dn**

**[Function]**

본 명령은, XB 및 YB에 저장 되어 있는 Address와 Dn에 나타나는 Data의 논리(OR)를 취한 Address를 XC(XS)(XK) 및 YC(YS)(YK)에 저장할 경우에 사용해 주세요.

**[Format]**

**XC<XB'Dn YC<YB'Dn**

**XS<XB'Dn YS<YB'Dn**

**XK<XB'Dn YK<YB'Dn**

(Dn=D1A ~ D1H, D2A ~ D2D, D3, D4)

**[Description]**

**[Exercise]**

**▶** **C[S][K]< B'Dn**

**[Function]**

본 명령은, XB 및 YB에 저장된 Address와 Dn에 나타나는 Bit를 반전한(XOR) Address를 발생시킵니다.

**[Format]**

**XC<XB"Dn** XBDn→XC

**XS<XB"Dn** XBDn→XS

**XK<XB"Dn** XBDn→XK

**YC<YB"Dn** YBDn→YC

**YS<YB"Dn** YBDn→YS

**YK<YB"Dn** YBDn→YK

(Dn=D1A ~ D1H, D2A ~ D2D, D3, D4)

: 배타적 논리합(XOR)

**[Description]**

본 명령을 이용한 Bit 변화 Pattern

Decoder 회로에서 오 선택이나 다중 선택에 의한 불량을 효율적으로 검출하기 위한 Test Pattern입니다.

**[Exercise]**

다음 pattern 순서는 Row Address Decoder를 Test하는 경우입니다.

► Pattern 순서

1. 전Cell에"0"을 쓴다.

2. 주목 Cell에"0"을 쓴다.

3. 주목 Cell의 Address와 1 bit 반전한 Address에"1"을 쓴다.

4. 주목 Cell의"0"을 읽어낸다.

5. 반전하는 Bit를 바꾸어 2 ~ 4항을 반복한다.

6. 주목 Cell을 바꾸어 2 ~ 5항을 반복한다.

▪Test Program

|  |
| --- |
| MPAT BIT ;ROW  REGISTER  XMAX=#7 F  YMAX=#7 F  TPH=#0  DCMR=#0  PC=#0  D3B=#1  IDX1=#3 FFE  IDX2=#3  IDX3=#7 E  START #0  NOP XB<0 YB<0 TP<TPH D3<D3B  LB0: JNI1 LB0 W XB<XB+1 YB<YB+1^BX X<XB Y<YB  LB1: NOP W XC<XB"D3 X<XB Y<YB  NOP W X<XC Y<YB /D  JNI1 #3 LB1 R X<XB Y<YB D3<D3\*2  NOP W XC<XB"D3 X<XB Y<YB  NOP W X<XC Y<YB /D  JNI1 #7E LB1 R XB<XB+1 X<XB Y<YB D3<D3B  NOP  STPS  END |

* + - **C[S][K]<f (C[S][K])**

**[Function]**

C[S][K]<f(C[S][K])는, XC(XS)(XK)및 YC(YS)(YK)의 Register를 이용해 Disturb Cell의 Address 발생을 연산에 의해서 얻는 명령 군입니다.

**[Format]**

|  |
| --- |
| C<C, S<S또는K<K  C[S][K]<C[S][K]±1  C[S][K]<Dn  C[S][K]<0  C[S][K]<C[S][K]\*2 🡨Shift left  C[S][K]</C[S][K] 🡨Complement  C[S][K]<C[S][K]&Dn 🡨논리적  C[S][K]<C[S][K]"Dn 🡨배타적 논리합  C[S][K]<C[S][K]'Dn 🡨논리합  C[S][K]<C[S][K]±1^BX  C[S][K]<C[S][K]±Dn^BX  C[S][K]<C[S][K]±1±BX  C[S][K]<C[S][K]±Dn±BX 🡨YC또는, YS,YK 에서만 유효  C[S][K]<C[S][K]±1^CY  C[S][K]<C[S][K]±Dn^CY  C[S][K]<C[S][K]±1±CY  C[S][K]<C[S][K]±Dn±CY |

C는 XC 또는 YC를 나타냅니다.

S는 XS 또는 YS를 나타냅니다.

K는 XK 또는 YK를 나타냅니다.

C[S][K]는 XC, XS 또는 XK 및 YC, YS 또는 YK를 나타냅니다.

Dn는 D1A ~ D1H, D2A ~ D2D, D3, D4를 나타냅니다.

▶C<C, S<S또는K<K

**[Function]**

본 명령은, Hold 명령으로 아무것도 실행하지 않을 때에 사용해 주세요. 본 명령은 생략 가능 합니다.

**[Format]**

**XC<XC YC<YC**

**XS<XS YS<YS**

**XK<XK YK<YK**

**[Description]**

**[Exercise]**

▶ C[S][K]<C[S][K]±1

**[Function]**

본 명령은, Address의 Increment 또는, Decrement에 사용해 주세요.

**[Format]**

**XC<XC(XS)(XK)±1 YC<YC(YS)(YK)±1**

**XS<XC(XS)(XK)±1 YS<YC(YS)(YK)±1**

**XK<XC(XS)(XK)±1 YK<YC(YS)(YK)±1**

**[Description]**

**[Exercise]**

본 명령어로 다음과 같은 순서를 만들어 낼 수 있습니다.

Ex1.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

0 X XMAX

0

Y

YMAX

XC<XC+1

(XS<XS+1)

(XK<XK+1)

Ex2.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

0 X XMAX

0

Y

YMAX

XC<XC-1

(XS<XS-1)

(XK<XK-1)

Ex3.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

0 X XMAX

0

Y

YMAX

YC<YC+1

(YS<YS+1)

(YK<YK+1)

Ex4.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

0 X XMAX

0

Y

YMAX

YC<YC-1

(YS<YS-1)

(YK<YK-1)

Ex5.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

0 X XMAX

0

Y

YMAX

XC<XC+1

YC<YC+1

XS<XS+1

YS<YS+1

XK<XK+1

YK<YK+1

Ex6.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

0 X XMAX

0

Y

YMAX

XC<XC-1

YC<YC-1

XS<XS-1

YS<YS-1

XK<XK-1

YK<YK-1

▶ C[S][K]<C[S][K]±Dn

**[Function]**

본 명령은, XC(XS)(XK), YC(YS)(YK)에 저장되어 있는 Disturb Address의 가산 및 감산을 실시하는 경우에 사용해 주세요

**[Format]**

**XC<XC(XS)(XK)+Dn YC<YC(YS)(YK)+Dn**

**XS<XC(XS)(XK)+Dn YS<YC(YS)(YK)+Dn**

**XK<XC(XS)(XK)+Dn YK<YC(YS)(YK)+Dn**

C+Dn<MAX+1일 때 (C+Dn) ->C

C+Dn>MAX 일 때 (C+Dn-MAX) →C

**XC<XC(XS)(XK)-Dn YC<YC(YS)(YK)-Dn**

**XS<XC(XS)(XK)-Dn YS<YC(YS)(YK)-Dn**

**XK<XC(XS)(XK)-Dn YK<YC(YS)(YK)-Dn**

C-Dn ≥0 일 때 (C-Dn) →C

C-Dn ≥0 일 때 XMAX+1-(C-Dn) →C

MAX는, XMAX, YMAX를 나타냅니다.

(Dn=D1A ~ D1H, D2A ~ D2D, D3, D4)

**[Description]**

본 명령은, 아래와 같은 Address 발생을 실시할 때 사용해 주세요.

**XC<XC-n, YC<YC-n**

**XC<XC, YC<YC-n**

**XC<XC+n, YC<YC-n**

**XC<XC-n, YC<YC**

**XC<XC+n, YC<YC**

**XC<XC-n, YC<YC+n**

**XC<XC, YC<YC+n**

**XC<XC+n, YC<YC+n**

**XC<XC, YC<YC**

n은 Dn을 나타낸다.

**[Exercise]**

► **XC<XC+D3 YC<YC+1^CY**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

0 X XMAX 0 X XMAX

0

X

YMAX

0

Y

YMAX

|  |
| --- |
| MPAT SAMPLE  REGISTER  XMAX=#7  YMAX=#7  D3=#3 연산 수  IDX1=#14  PC=#0  START #0  NOP XB<0 YB<0  NOP XC<XB YC<YB ; XC, YC의 Initialize  LB0: JLN1 LB0 XC<XC+D3 YC<YC+1^CY X<XC Y<YC  :  END |

▶ C[S][K]< C[S][K]±1, Dn^BX

**[Function]**

본 명령은, XB의 연산에 대해 Carry 또는 Borrow-가 발생했을 때의 조건부 연산 명령입니다.

**[Format]**

**YC<YC(YS)(YK) ±1^BX**

**YS<YC(YS)(YK) ±1^BX**

**YK<YC(YS)(YK) ±1^BX**

XB Carry/Borrow-가 발생하지 않는 경우 YC(YS)(YK)<YC(YS)(YK)

XB Carry/Borrow-가 발생했을 경우 YC(YS)(YK)<YC(YS)(YK)±1

**YC<YC(YS)(YK) ± Dn^BX**

**YS<YC(YS)(YK) ± Dn^BX**

**YK<YC(YS)(YK) ± Dn^BX**

XB Carry/Borrow-가 발생하지 않는 경우 YC(YS)(YK)<YC(YS)(YK)

XB Carry/Borrow-가 발생했을 경우 YC(YS)(YK)<YC(YS)(YK)±Dn

**[Description]**

**[Exercise]**

▶ C[S][K]< C[S][K]±1, Dn±BX

**[Function]**

YC 및 YS, YK의 연산 명령에 대하고, XB의 연산으로 발생한 Carry를 가산, 감산 하는 명령입니다.

**[Format]**

**YC<YC(YS)(YK) ±1±BX**

**YS<YC(YS)(YK) ±1±BX**

**YK<YC(YS)(YK) ±1±BX**

XB Carry가 발생하지 않는 경우 YC(YS)(YK)<YC(YS)(YK)±1

XB Carry가 발생했을 경우 YC(YS)(YK)<YC(YS)(YK)±2

**YC<YC(YS)(YK) ± Dn±BX**

**YS<YC(YS)(YK) ± Dn±BX**

**YK<YC(YS)(YK) ± Dn±BX**

XB Carry가 발생하지 않는 경우 YC(YS)(YK)<YC(YS)(YK)±Dn

XB Carry가 발생했을 경우 YC(YS)(YK)<YC(YS)(YK)±Dn±1

**[Description]**

**[Exercise]**

▶ C[S][K]< C[S][K]±1, Dn^CY

**[Function]**

본 명령은, XC(XS)(XK)와 YC(YS)(YK)의 각 Address의 연결 명령입니다.

**[Format]**

**YC<YC(YS)(YK) ±1^CY**

**YS<YC(YS)(YK) ±1^CY**

**YK<YC(YS)(YK) ±1^CY**

XC(XS)(XK) Carry가 발생하지 않는 경우 YC(YS)(YK)<YC(YS)(YK)

XC(XS)(XK) Carry가 발생했을 경우 YC(YS)(YK)<YC(YS)(YK)±1

**YC<YC(YS)(YK) ± Dn^CY**

**YS<YC(YS)(YK) ± Dn^CY**

**YK<YC(YS)(YK) ± Dn^CY**

XC(XS)(XK) Carry가 발생하지 않는 경우 YC(YS)(YK)<YC(YS)(YK)

XC(XS)(XK) Carry가 발생했을 경우 YC(YS)(YK)<YC(YS)(YK)±Dn

**[Description]**

XC Carry가 발생했을 경우 발생할 수 있는 pattern

XC<XC+m

YC<YC-n^CY

XC<XC+n

XC<XC+m

YC<YC+1^CY

XC<XC+m

YC<YC+n^CY

n, m는, D1A ~ D1H, D2A ~ D2D, D3, D4를 나타냅니다.

m는 XC+m의 연산 결과가 LMAX<XC+m의 조건을 채우는 값을 나타냅니다.

XC<XC+m

YC<YC-n^CY

XC<XC+m

YC<YC+1^CY

XC<XC-n

XC<XC-m

YC<YC+n^CY

n, m는, D1A ~ D1H, D2A ~ D2D, D3, D4를 나타냅니다.

m는 XC-m연산 결과가 0>XC-m의 조건을 채우는 값을 나타냅니다.

**[Exercise]**

▶ C[S][K]<C[S][K]±1, Dn±CY

**[Function]**

본 명령은, YC, YS 및 YK의 연산 명령에 대한 것이고, XC, XS 및 XK의 연산으로 발생한 Carry를 가산, 감산하는 명령입니다. 본 명령은, Shift diagonal등의 Disturb Address의 발생에 사용해 주세요.

**[Format]**

**YC<YC(YS)(YK) ±1±CY**

**YS<YC(YS)(YK) ±1±CY**

**YK<YC(YS)(YK) ±1±CY**

XC(XS)(XK) Carry가 발생하지 않는 경우 YC(YS)(YK)<YC(YS)(YK) ±1

XC(XS)(XK) Carry가 발생했을 경우 YC(YS)(YK)<YC(YS)(YK)±2

**YC<YC(YS)(YK) ± Dn±CY**

**YS<YC(YS)(YK) ± Dn±CY**

**YK<YC(YS)(YK) ± Dn±CY**

XC(XS)(XK) Carry가 발생하지 않는 경우 YC(YS)(YK)<YC(YS)(YK) ±Dn

XC(XS)(XK) Carry가 발생했을 경우 YC(YS)(YK)<YC(YS)(YK)±Dn±1

**[Description]**

**[Exercise]**

본 명령은, Shift diagonal등의 Disturb Address의 발생에 사용해 주세요.

ST0 : JNI1 ST0 XC<XC+1 YC<YC+1+CY X<XC Y<YC

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 14 | 11 | 8 |
| 5 | 2 | 15 | 12 |
| 9 | 6 | 3 | 16 |
| 13 | 10 | 7 | 4 |

(HMAX)

Y

(LMAX)

X

(LMAX)

X

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 14 | 11 | 8 |
| 5 | 2 | 15 | 12 |
| 9 | 6 | 3 | 16 |
| 13 | 10 | 7 | 4 |

Y

(HMAX)

XC Carry =1일 때(4, 8, 12, 16)마다 YC<YC+2를 실행하고, 다른Step에서는, YC<YC+1을 실행합니다

ST0 : JNI1 ST0 XC<XC-1 YC<YC-1-CY X<XC Y<YC

HMAX

Y

(LMAX)

X

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | 7 | 10 | 13 |
| 5 | 3 | 6 | 9 |
| 12 | 6 | 2 | 5 |
| 8 | 11 | 7 | 1 |

XC Borrow -=1일 때(좌도 4, 8, 12, 16)마다 YC<YC-2를 실행하고, 다른 Step에서는YC<YC-1을 실행합니다.

▶ C[S][K]<Dn

**[Function]**

XC(XS)(XK), YC(YS)(YK)의 각 Register를 Initialize 하는 명령입니다.

**[Format]**

**XC<Dn YC<Dn**

**XS<Dn YS<Dn**

**XK<Dn YK<Dn**

Dn (D1A ~ D1H, D2A ~ D2D, D3, D4) Register의 내용이 Load 됩니다.

**[Description]**

**[Exercise]**

▶ C[S][K]<0

**[Function]**

XC(XS)(XK), YC(YS)(YK)의 각 Register를 Clear(#0) 하는 명령입니다.

**[Format]**

**XC<0 YC<0**

**XS<0 YS<0**

**XK<0 YK<0**

**[Description]**

**[Exercise]**

▶ C[S][K]<C[S][K]\*2

**[Function]**

XC(XS)(XK), YC(YS)(YK)의 각 Register로 지정된 Register의 내용을 Left shift(LSB=0)하고 Load하는 명령입니다.

**[Format]**

**XC<XC(XS)(XK) \*2 YC<YC(YS)(YK) \*2**

**XS<XC(XS)(XK) \*2 YS<YC(YS)(YK) \*2**

**XK<XC(XS)(XK) \*2 YK<YC(YS)(YK) \*2**

**[Description]**

**[Exercise]**

▶ C[S][K]</C[S][K]

**[Function]**

XC(XS)(XK), YC(YS)(YK)의 각 Register로 지정된 Register의 내용을 반전하고 Load하는 명령 입니다.

**[Format]**

**XC</XC(XS)(XK) YC</YC(YS)(YK)**

**XS</XC(XS)(XK) YS</YC(YS)(YK)**

**XK</XC(XS)(XK) YK</YC(YS)(YK)**

**[Description]**

**[Exercise]**

▶ C[S][K]<C[S][K]&Dn

**[Function]**

XC(XS)(XK), YC(YS)(YK)의 각 Register로 지정된 XC(XS)(XK), YC(YS)(YK)의 Register내용과 Dn에 나타난 Data와의 논리곱(AND)을 취해 Load하는 명령입니다.

**[Format]**

**XC<XC(XS)(XK)&Dn YC<YC(YS)(YK)&Dn**

**XS<XC(XS)(XK)&Dn YS<YC(YS)(YK)&Dn**

**XK<XC(XS)(XK)&Dn YK<YC(YS)(YK)&Dn**

**[Description]**

**[Exercise]**

▶ C[S][K]<C[S][K]"Dn

**[Function]**

XC(XS)(XK), YC(YS)(YK)의 각 Register로 지정된 XC(XS)(XK), YC(YS)(YK)의 Register내용과 Dn로 나타난 Data와의 배타적 논리합(**XOR**)을 취해 Load하는 명령입니다.

**[Format]**

**XC<XC(XS)(XK) "Dn YC<YC(YS)(YK) "Dn**

**XS<XC(XS)(XK) "Dn YS<YC(YS)(YK) "Dn**

**XK<XC(XS)(XK) "Dn YK<YC(YS)(YK) "Dn**

**[Description]**

**[Exercise]**

▶ C[S][K]'Dn

**[Function]**

XC(XS)(XK), YC(YS)(YK)의 각 Register로 지정된 XC(XS)(XK), YC(YS)(YK)의 Register내용과 Dn로 나타난 Data와의 논리합(**OR**)을 취해 Load하는 명령입니다.

**[Format]**

**XC<XC(XS)(XK) 'Dn YC<YC(YS)(YK) 'Dn**

**XS<XC(XS)(XK) 'Dn YS<YC(YS)(YK) 'Dn**

**XK<XC(XS)(XK) 'Dn YK<YC(YS)(YK) 'Dn**

**[Description]**

**[Exercise]**

* + - **C[S][K]<f(B,C[S][K],t)**

**[Function]**

ALPG는, 주소 순서가 복잡한 Test Pattern을 발생시키기 위해서 XC, YC의 각 주소를 명령으로부터 임시대피 하기 위한 Save Register XS, XK, YS, YK를 가지고 있습니다.

이러한 Save Register를 제어하기 위해서 아래와 같은 명령이 있습니다.

**[Format]**

**C<S**

**S<C**

**C<>S**

**CS<OPE**

**C<S<OPE**

**S<C<OPE**

**C<K**

**K<C**

**C<>K**

**CK<OPE**

**C<K<OPE**

**K<C<OPE**

**SK<OPE**

**CSK<OPE**

**[Memo]**

C는 XC 또는 YC

S는, XS 또는 YS

CS는, XCS 또는 YCS

OPE는, f (B)와 f (C[S][K])의 각 연산

K는, XK 또는 YK

CK는, XCK 또는 YCK

SK는, XSK 또는 YSK

CSK는, XCSK 또는 YCSK

**[Description]**

* 각 statement 별 Mnemonic과 설명입니다.

|  |  |  |
| --- | --- | --- |
| Statement | Mnemonic | Description |
| C<S | XC<XS, YC<YS | Save Register에 임시저장되고 있던 Address를 XC, YC Register에 Load하는 명령입니다. |
| S<C | XS<XC, YS<YC | XC, YC Register의 Address를 Save Register에 임시저장 하는 명령입니다. |
| C<>S | XC<>XS, YC<>YS | Save Register XS, YS에 저장되고 있는 Address와 XC, YC Register에 저장되어 있는 Address를 바꾸는 명령입니다. |
| CS<OPE | XCS<OPE, YCS<OPE | XC, YC의 각 연산 명령에 대해, XS, YS에 동시에 연산 결과를 저장하는 명령입니다. |
| C<S<OPE | XC<XS<OPE, YC<YS<OPE | XS, YS의 각 Save Register에 저장되어 있는 Address를 XC, YC의 각 Register에 Load하고, 연산 결과를 XS, YS의 각 Save Register에만 저장하는 명령입니다.  연산에 사용되는 XC(XS), YC(YS)의 각 Address는 Load 되기 전의 Address가 사용됩니다. |
| S<C<OPE | XS<XC<OPE, YS<YC<OPE | XC, YC의 각 Register에 저장되고 있는 Address를 XS, YS의 각 Save Register에 Load하고, 연산 결과를 XC, YC의 각 Register에만 저장하는 명령입니다.  연산에 사용되는 XC(XS), YC(YS)의 각 Address는, Load 되기 전의 Address가 사용됩니다. |
| C<K | XC<XK, YC<YK | Save Register XK, YK에 임시 저장되고 있던 Address를 XC, YC Register에 Load하는 명령입니다. |
| K<C | XK<XC, YK<YC | XC, YC Register의 Address를 Save Register XK, YK에 임시 저장하는 명령입니다. |
| C<>K | XC<>XK, YC<>YK | Save Register XK, YK 에 저장되고 있는 Address와 XC, YC Register에 되어있는 Address를 바꾸는 명령입니다. |
| CK<OPE | XCK<OPE, YCK<OPE | XC, YC의 각 연산 명령에 대하고, XK, YK에 동시에 연산 결과를 저장하는 명령입니다. |
| C<K<OPE | XC<XK<OPE, YC<YK<OPE | XK, YK의 각 Save Register에 저장되고 있는 Address  를 XC, YC의 각 Register에 Load하고, 연산 결과를 XK, YK의 각 Save Register에만 저장하는 명령입니다.  연산에 사용되는 XC(XK), YC(YK)의 각 Address는 Load 되기 전의 Address가 사용됩니다. |
| K<C<OPE | XK<XC<OPE, YK<YC<OPE | XC, YC의 각 Register에 저장되고 있는 Address를 XK, YK의 각 Save Register에 Load하고, 연산 결과를 XC, YC의 각 Register에 저장하는 명령입니다.  연산에 사용되는 XC(XK), YC(YK)의 각 Address는, Load 되기 전의 Address가 사용됩니다. |
| SK<OPE | XSK<OPE, YSK<OPE | XC, YC의 각 연산 명령에 대한 연산 결과를 XS, XK, YS, YK에 동시에 저장하는 명령입니다. |
| CSK<OPE | XCSK<OPE, YCSK<OPE | XC, YC의 각 연산 명령에 대하고, 연산 결과를 XC, XS, XK, YC, YS, YK에 동시에 저장하는 명령입니다. |

**[Exercise]**

* + **Source Address Field**

**[Function]**

DUT에 인가하는 주소를 선택하는 명령입니다.

**[Format]**

**X<XB Y<YB**

**X<XC Y<YC**

**X<XS Y<YS**

**X<XK Y<YK**

**X<YB Y<XB**

**X<YC Y<XC**

**X<YS Y<XS**

**X<YK Y<XK**

**X<XB'Z Y<XB'Z**

**X<XC'Z Y<XC'Z**

**X<XS'Z Y<XS'Z**

**X<XK'Z Y<XK'Z**

**X<Z Y<Z**

**[Description]**

1. X<XB와 Y<YB는 생략할 수 있습니다. (기술하지 않는 경우 이 명령이 실행 됩니다.)

2. X Address 발생의 Register(XB, XC, XS, XK)의 그룹과 Y주소 발생의 Register (YB, YC, YS, YK)의 그룹에 대해, X, Y의 출력 선택의 편성으로서 같은 그룹 내가 다른 Register를 선택할 수 없습니다.

3. DUT에 인가하는 X, Y주소의 유효 Bit는, X주소는 XMAX Register, Y주소는 YMAX Register에 지정해 주세요. XMAX, YMAX는 유효 Bit수를 n로 했을 때, 2n-1으로 지정해 주십시오.

4. 아래에 본 명령의 세부 기능을 기술 하였다.

►주소 발생 순서가 X 향으로 주사 시 사용해 주세요

X<XB Y<YB

X<XC Y<YC

X<XS Y<YS

X<XK Y<YK

►주소 발생 순서 : Y 향으로 주사 시 사용해 주세요

X<YB Y<XB

X<YC Y<XC

X<YS Y<XS

X<YK Y<XK

►이 명령은 X의 Carry가 Y를, Y의 Carry가 X의 주소 발생을 컨트롤 하는 경우에 사용해 주세요

X<XB'Z Y<XB'Z

X<XC'Z Y<XC'Z

X<XS'Z Y<XS'Z

X<XK'Z Y<XK'Z

►이 명령은, 주로 주소 Line을 이용하고, 주소 이외의 Data를 DUT 로 설정할 경우에 사용 해주세요.

X<Z Y<Z

**[Description]**

* + **N Register Field**

**[Function]**

Bank 주소의 발생 등에 사용하는 N Register의 연산 명령입니다.

**[Format]**

N<N

N<N+1

N<N-1

N<NH

N<0

N</N

N<NDn

N<N+NDn

N<N-NDn

**[Description]**

► N<N

N Register의 HOLD 명령입니다. N Register 의 연산을 실시하지 않는 경우에 사용합니다.

본 명령을 포함한 N Register 연산 명령을 생략 하는 경우, 본 명령을 지정하는 것과 같게 됩니다.

► N<N+1

N Register의 내용을 Increment 하는 명령입니다.

► N<N-1

N Register의 내용을 Decrement 하는 명령입니다.

► N<NH

N Register에 NH Register의 내용을 Load 하는 명령입니다. N Register의 Initialize에 사용합니다..

► N<0

N Register의 내용을 0에 Clear 하는 명령입니다. N Register의 Initialize에 사용합니다.

► N</N

N Register의 내용을 반전해, 반전한 값을 N Register에 저장하는 명령입니다.

► N<NDn

N Register에 NDn Register의 내용을 Load하는 명령입니다. N Register의 Initialize에 사용합니다.

► N<N+NDn

N Register에 NDn Register의 내용을 가산하는 명령입니다.

► N<N-NDn

N Register로부터 NDn Register의 내용을 감산하는 명령입니다.

► NDn는, ND1 ~ ND7를 나타냅니다.

【 메모 】

ND1는 ND와 기술할 수 있습니다.

**[Exercise]**

* + **B Register Field**

**[Function]**

Bank 주소의 발생 등에 사용하는 B Register의 연산 명령입니다.

**[Format]**

**B<B**

**B<B+1**

**B<B-1**

**B<BH**

**B<0**

**B</B**

**B<BDn**

**B<B+BDn**

**B<B-BDn**

**[Description]**

►B<B

B Register의 HOLD 명령입니다. B Register의 연산을 실시하지 않는 경우에 사용합니다.

본 명령을 포함한 B Register 연산 명령을 생략 하는 경우, 본 명령을 지정하는 것과 같습니다.

►B<B+1

B Register의 내용을 Increment 하는 명령입니다.

►B<B-1

B Register의 내용을 감소 하는 명령입니다.

►B<BH

B Register에 BH Register의 내용을 Load하는 명령입니다. B Register의 Initialize에 사용 합니다.

►B<0

B Register의 내용을 0에 Clear하는 명령입니다. B Register의 Initialize에 사용합니다.

►B</B

B Register의 내용을 반전해, 반전한 값을 B Register에 Store하는 명령입니다.

►B<BDn

B Register에 BDn Register의 내용을 Load하는 명령입니다. B Register의 Initialize에 사용합니다.

►B<B+BDn

B Register에 BDn Register의 내용을 가산하는 명령입니다.

►B<B-BDn

B Register로부터 BDn Register의 내용을 감산하는 명령입니다.

►BDn는, BD1 ~ BD7를 나타냅니다.

【 메모 】

BD1는 BD로 기술할 수 있습니다.

**[Exercise]**

* + **Address Inversion Field**

**[Function]**

DUT에 인가하는 주소를 반전하는 명령입니다. 기술한 Cycle만 X, Y, Z의 각 주소가 반전함, 출력됩니다.

**[Format]**

**/X** XMAXn ◦ Xn → Xn

**/Y** YMAXn ◦ Yn → Yn

**/Z** ZMAXn ◦ Zn → Zn

(n=-0~23) ◦ 은 or 기능을 하는 기호 입니다.

**[Description]**

**[Exercise]**

Ex1.

|  |
| --- |
| MPAT SAMPLE  REGISTER  XMAX=#E  YMAX=#F  ZMAX=#9  START #0  NOP XB<0 YB<0 Z<0 ; X Y Z  NOP X<XB Y<YB Z<Z ;#0 #0 #0  NOP X<XB Y<YB Z<Z /X /Y /Z ;#E #F #9  NOP X<XB Y<YB Z<Z ;#0 #0 #0 |

Ex2.Marching pattern

|  |
| --- |
| MPAT MARCH  REGISTER  XMAX=#7F  YMAX=#7F  TPH=#0  DCMR=#0  IDX1=#3FFE  PC=#0  START #00  NOP XB<0 YB<0 TP<TPH  A  B  B  C  LB0: JNI1 LB0 W XB<XB+1 YB<YB+1^BX X<XB Y<YB /D  LB1: NOP R X<XB Y<YB /D  JNI1 LB1 W XB<XB+1 YB<YB+1^BX X<XB Y<YB  LB2: NOP R X<XB Y<YB /X /Y  JNI1 LB2 W XB<XB+1 YB<YB+1^BX X<XB Y<YB /X /Y /D  NOP  STPS  END |

R1W0

R1W0

R1W0

R1W0

R0W1

R1W0

R1W0

R1W0

|----------A-----|--------B----------|---------C------|

* + **D3, D4 Register Field**

**[Function]**

XC, YC에 저장되는 Disturb Address를 발생할 경우에 사용되는 연산 수, 초기치를 저장합니다.

**[Format]**

**D3<D3 D4<D4**

**D3<D3\*2 D4<D4\*2**

**D4<D4/2**

**D3<D3B D4<D4B**

**D3<D3\*2L D4<D4\*2L**

**D3<D3+1**

**D3<D3-1**

**D3<0**

**[Description]**

►D3<D3B, D4<D4B

본 명령은, D3, D4Register를 초기화하는 명령으로, 각각 D3B, D4B Register의 값이 Load 됩니다.

►D3<D3+1

본 명령은, D3 Register의 값을 Increment 하는 명령입니다.

►D3<D3-1

본 명령은, D3Register의 값을 Decrement하는 명령입니다.

►D3<D3\*2, D4<D4\*2

본 명령은, D3, D4Register의 Shift 명령으로, 아래와 같은 동작을 실시합니다.

►D4<D4/2

본 명령은, D4Register의 Shift 명령으로, 아래와 같은 동작을 실시합니다.

►D3<D3\*2L D4<D4\*2L

본 명령은, D3Register와 D4Register를 연결해 Shift 하는 명령입니다. LMAX로 나타난 D3 Register의 유효 Bit의 MSB로부터 Shift 아웃 하는 Data가 D4Register의 LSB의 Data로서 Load 됩니다.

►D3<0

본 명령은, D3Register를 Initialize 하는 명령으로 D3Register는 0(Clear)이 됩니다.

**[Exercise]**

* + **Address Swap Field**

**[Function]**

일반적으로 LMAX는 XB, XC, XS, XK, D3 Register의 연산의 유효 bit제어를 실시하고, HMAX는 YB, YC, YS, YK, D4 Register의 연산을 실시합니다. 이 명령으로 인해 LMAX와 HMAX에 의해 제어되는 Register의 값이 교체됩니다.

**[Format]**

**LMAX<>HMAX**

**[Description]**

이 명령은 X Address와 Y Address의 bit length가 다른 DUT의 Test에 있어서, RealTime으로 주사방향을 바꿀 때 사용해 주세요.

(LMAX) (LMAX)

X XMAX X XMAX

(HMAX) (HMAX)

YMAX YMAX

X Address 방향 주사 Y Address 방향 주사

**[Exercise]**

* + **Program 작성시 주의점**

► XMAX, YMAX, ZMAX

DUT에 인가하는 X, Y 및 Z주소의 유효 Bit의 지정을 XMAX, YMAX, ZMAX로 지정해 주세요.

► LMAX, HMAX

YB, YC, YS, YK, Z의 연산에 대하고, 조건부 연산 명령을 사용할 때는 LMAX, HMAX를 지정해 주세요. LMAX 및 HMAX의 지정이 없을 때는 다음의 값이 지정됩니다.

LMAX=XMAX

HMAX=YMAX

► 주소 연산의 Default

주소 연산 명령의 Default를 이하에 나타냅니다.

XB<XB YB<YB XC<XC XS<XS XK<XK YC<YC YS<YS YK<YK

D3<D3 D4<D4 N<N Z<Z X<XB Y<YB B<B

# **DATA PATTERN 발생 명령**

* + **DATA 발생부의 Register**

Data Register는 Data 발생에 관련된 Register들입니다.

각 Register의 설정과 관련해서는 아래 표1-1 Data Control Register List를 참고 하십시오.

★표 5-1 Data Register List

|  |  |  |  |
| --- | --- | --- | --- |
| Register | Description | Bits | Set |
| TP, TP2 | DUT에 인가하는 Data 및 DUT로부터 출력되는 Data와 비교 Data(기대치)의 발생에 사용 됩니다. | 48 | R/  W |
| TPH, TPH2 | TP, TP2 Register에 Load하는 초기치를 저장하는 Register입니다. | 48 | R/  W |
| DCMR, DCMR2 | Data Pattern 발생 명령의/D 및/D2에 의해서 반전시키는 Bit의 마스크를 지정하는 Register 입니다. 본Register의"0"의 Bit에 대해서, /D 및/D2의 기술된 Cycle로 Data의 논리반전을 합니다  (Default : DCMR=#0, DCMR2=#0) |  |  |
| DFLG | DFLG Register가"1"때, TP Register의 반전한 값이 DA0-23에 출력됩니다.  DFLG Register는, JZD 명령에 의해서 반전합니다.  DFLG에 의해 TP2 Register의 반전한 값을 DB0-23에 출력하기 위해서는 MODE문으로 TP2INV를 지정해 주세요. | 24 | R/  W |
| FP Generator | FP Generator는, X와 Y의 주소를 이용한 주소 함수로, 이 주소 함수의 출력이"1"인 것과, TP Register의 반전한 값이 DA0-23에 출력됩니다.  FP Generator에 의해 TP2 Register의 반전한 값을 DB0-17에 출력하려면 MODE문으로 TP2INV를 지정해 주세 | 24 | R/  W |
| XMASK, YMASK | 주소 함수로서 사용하는 X, Y의 유효 Bit를 지정하는 Register입니다. | 24 | R/  W |
| DSD, RCD, CCD | 주소 함수의 Offset치 또는 비교치로서 사용하는 Register입니다. | 24 | R/  W |
| D5, D6 | D5 는 TP, D6 는 TP2 와의 Data 연산을 실시할 때 사용합니다. | 48 | R/  W |
| ARIRAM | ARIRAM에 기술된 특정의 주소로 Pattern Data(DA0-23)를 반전할 경우에 사용합니다.  ARIRAM에 의해 TP2 Register의 반전한 값을 DB0-17에 출력하려면 , MODE문으로 TP2INV를 지정해 주세요. |  | R/  W |
| Parity Generator | MODE문에 의해서 PARITY가 지정되어 있을 때, DA0-11의 홀수 패리티를 DA24에 DA12-23의 홀수 패리티를 DA25에 출력하는 것입니다. |  | R/  W |

**[Memo]**

►

이 Register는 SET 명령, DSET 명령, Test Plan program 및 Test Pattern Program에 따라 설정할 수있다.

* + **Data 발생 명령의 구성**

**[ 개요 ]**

Data 발생 부는, 다음의 각 명령으로 구성되어 있습니다.

► TP, TP2 register field

► FP function field

► DSD, RCD, CCD field

► Data inversion field

► Data hold field

► Source data field

► BWPE field

**[ Description ]**

1. Data 발생 명령부의 각 명령의 Mnemonic을 아래에 기술 하였습니다.

2. 괄호로 묶여진 Group내에서 동일 Cycle에는 1개의 명령선택이 가능 합니다.

3. 사각형은 1개의 명령 선택을 의미 합니다.

**[ Mnemonic ]**

**▶TP, TP2 Register Field**

TP는 TP1, TPS는 TPS1, TPH는 TPH1와 기술할 수 있습니다.

TP</TP

TP<TP\*2

TP<TP/2

TP</TP

TP<0

TP<TP+1

TP<TP-1

TP<TPH

TP<D5

TP<TP'D5 (OR)

TP<TP&D5 (AND)

TP<TP"D5 (XOR)

TP<TP+D5

TP<TP-D5

TP</D5

TP<TPS

TPS<TP

TP<>TPS

TP2<TP2

TP2<TP2\*2

TP2<TP2/2

TP2</TP2

TP2<0

TP2<TP2+1

TP2<TP2-1

TP2<TPH2

TP2<D6

TP2<TP2'D6 (OR)

TP2<TP2&D6 (AND)

TP2<TP2"D6 (XOR)

TP2<TP2+D6

TP2<TP2-D6

TP2</D6

TP2<TPS2

TPS2<TP2

TP2<>TPS2

**▶FP Function Field**

FP0

FP1

FP3

FP7

FP8

FP9

FPX

(Fix 0)

(Checker Board)

(Diagonal)

(Mask Parity)

(Row Bar)

(Column Bar)

(Universal FP)

**▶DSD, RCD, CCD Field**

DSD<0

RCD<0

CCD<0

DSD<DSD

RCD<RCD

CCD<CCD

DSD<DSD+1

RCD<RCD+1

CCD<CCD+1

RCD<RCD+1 CCD<CCD+1

**▶ Data Inversion Field**

[ /D ]

[ /D2 ]

**▶ Data Hold Field**

[ PD ]

**▶ Source Data Field**

[ D<TP1 ]

[ D<TP2 ]

[ D<TPXOR ]

TPXOR은 TP1.XOR.TP2 입니다.

* + **TP, TP2 Register Field**

**[Function]**

DUT에 인가할 Data Pattern 및 기대치 Data Pattern을 TP, TP2 Register의 연산에 의해서 발생시키는 명령입니다.

**[Format ]**

**TP<TP TP2<TP2 (Hold)**

**TP<TPH TP2<TPH2 (Load)**

**TP<TP\*2 TP2<TP2\*2 (Shift left)**

**TP<TP/2 TP2<TP2/2 (Shift Right)**

**TP<TP+1 TP2<TP2+1 (Increment)**

**TP<TP-1 TP2<TP2-1 (Decrement)**

**TP<0 TP2<0 (Clear)**

**TP</TP TP2</TP2 (Compliment)**

**TP<D5 TP2<D6**

**TP<TP'D5 TP2<TP2'D6**

**TP<TP&D5 TP2<TP2&D6**

**TP<TP"D5 TP2<TP2"D6**

**TP<TP+D5 TP2<TP2+D6**

**TP<TP-D5 TP2<TP2-D6**

**TP</D5 TP2</D6**

**TPS<TP TPS2<TP2 (Save)**

**TP<TPS TP2<TPS2 (Recovery)**

**TP<>TPS TP2<>TPS2 (Swap)**

**[Description]**

▶TP<TP TP2<TP2 (Hold)

본 명령은, TP, TP2 Register의 HOLD 명령입니다. 본 명령의 기술은 생략 가능합니다.

▶TP<TPH TP2<TPH2 (Load)

본 명령은, TP, TP2 Register를 Initialize 하는 명령으로, TPH, TPH2 Register의 내용이 TP, TP2 Register에 Load 됩니다.

▶TP<TP\*2 TP2<TP2\*2 (Shift left)

본 명령은, TP, TP2 Register의 내용을 1Bit왼쪽 Shift 하고 TP, TP2 Register에 저장하는 명령입니다.

LSB 측에는 항상 0이 Load 됩니다. (Rotate shift가 아닙니다.)

▶TP<TP/2 TP2<TP2/2 (Shift Right)

본 명령은, TP, TP2 Register의 내용을 1Bit 오른쪽 Shift 해 TP, TP2 Register에 저장하는 명령입니다.

MSB측에는 항상 0이 Load 됩니다. (Rotate shift가 아닙니다.)

▶TP<TP+1 TP2<TP2+1 (Increment)

본 명령은, TP, TP2 Register의 내용을 Increment(+1) 해 TP, TP2 Register에 저장하는 명령입니다.

▶TP<TP-1 TP2<TP2-1 (Decrement)

본 명령은, TP, TP2 Register의 내용을 Decrement (-1) 해 TP, TP2 Register에 저장하는 명령입니다.

▶TP<0 TP2<0 (Clear)

본 명령은, TP, TP2 Register를 0에 Initialize 하는 명령입니다.

▶TP</TP TP2</TP2 (Compliment)

본 명령은, TP, TP2 Register의 내용을 반전시켜 저장하는 명령입니다.

▶TP<D5 TP2<D6

본 명령은, TP Register에 D5, TP2 Register에 D6의 내용을 Load하는 명령입니다.

▶TP<TP'D5 TP2<TP2'D6

본 명령은, TP Register와 D5, TP2 Register와 D6와의 논리합(OR)을 취해 TP, TP2 Register에 저장 하는 명령입니다.

▶TP<TP&D5 TP2<TP2&D6

본 명령은, TP Register와 D5, TP2 Register와 D6와의 논리곱(AND)을 취해 TP, TP2 Register에 저장 하는 명령입니다.

▶TP<TP"D5 TP2<TP2"D6

본 명령은, TP Register와 D5, TP2 Register와 D6와의 배타적 논리합(XOR)을 취 해 TP, TP2 Register에 저장하는 명령입니다.

▶TP<TP+D5 TP2<TP2+D6

본 명령은, TP Register와 D5, TP2 Register와 D6를 산술 가산해 TP, TP2 Register 에 저장하는 명령입니다.

▶TP<TP-D5 TP2<TP2-D6

본 명령은, TP Register로부터 D5, TP2 Register로부터 D6를 산술 감산해 TP, TP2 Register에 저장하는 명령입니다.

▶TP</D5 TP2</D6

본 명령은, TP Register에 D5의 반전치, TP2 Register에 D6의 반전치를 저장하는 명령입니다.

▶TPS<TP TPS2<TP2 (Save)

본 명령은, TPS, TPS2 Register에 TP, TP2 Register의 내용을 Load하는 명령입니다.

▶TP<TPS TP2<TPS2 (Recovery)

본 명령은, TP, TP2 Register에 TPS, TPS2 Register의 내용을 Load하는 명령입 니다.

▶TP<>TPS TP2<>TPS2 (Swap)

본 명령은, TP Register와 TPS, TP2 Register와 TPS2 Register의 내용을 바꿔 넣는 명령입니다.

**[Exercise]**

* + **FP Function Field**

**[Function]**

FP function field(FP함수)는, DUT에 인가하는 Pattern Data 및 기대치 Data를, DUT에 인가되는 X, Y주소의 논리 연산에 의해서 발생하는 명령입니다

**[Format ]**

**FP0 (Fix 0)**

**FP1 (Checker Board)**

**FP3 (Diagonal)**

**FP7 (Mask Parity)**

**FP8 (Row Bar)**

**FP9 (Column Bar)**

**FPX (Universal FP)**

**[Description]**

본 명령으로 선택한 함수의 출력이"1"인 것과, TP Register의 내용을 반전합니다. FP함수에 의해, TP2 Register의 내용을 반전하는 경우, MODE 문장으로 TP2INV를 지정해 주세요.

PRESCRAM를 사용한 배속 Mode를 지정했을 경우, TP1, TP2 Data에 대한 반전 조건 으 로서 배속 Mod~~e의 전반용과 후반용의 어느 쪽의 Y주소를 사용할까는, MODE문의DBFORMn (n=1 ~ 4)에서 지정할 수 있습니다.~~

이하에 각 함수와 선택 명령을 나타냅니다.

▶FP0 (Fix 0)

본 명령을 선택하면, TP(TP2) Register 및 X, Y주소에 관계없이, 항상"0"을 출력합니다.

▶FP1 (Checker Board)

본 명령을 선택하면, DUT 에 인가되는 X, Y 주소의 LSB 의 배타적 논리합의 조건에 따라서, TP(TP2) Register의 내용을 반전합니다.

▶FP3 (Diagonal)

본 명령을 선택하면, DUT에 인가되는 Y주소와 DSD Register의 값의 대수화가, X주소에 동일할 때에 TP(TP2) Register의 내용을 반전합니다.

또, 대수화와 X주소의 비교는, XMAX Register에 의해서 나타난 유효 Bit내에서 행 해집니다.

▶FP7 (Mask Parity)

본 명령을 선택하면, DUT에 인가되는 X, Y주소 가운데, XMASK, YMASK Register에서 지정된 Bit에 대해, "1"의 수가 홀수 때, TP(TP2) Register의 내용을 반전합니다.

▶FP8 (Row Bar)

본 명령을 선택하면, DUT에 인가되는 X주소로 XMASK Register가"0"의 Bit와 RCD내용을 비교해, 일치했을 때에 TP(TP2) Register의 내용을 반전합니다. (X.AND.XMASK)=RCD 일 때, TP(TP2) Register의 내용을 반전합니다.

▶FP9 (Column Bar)

본 명령을 선택하면, DUT에 인가되는 Y주소로 YMASK Register가"0"의 Bit와 CCD내용을 비교해, 일치했을 때에 TP(TP2) Register의 내용을 반전합니다. (Y.AND.YMASK) 일 때, TP(TP2) Register의 내용을 반전합니다.

▶FPX (Universal FP)

본 명령을 지정하는 Cycle에, FPSEL Register로 지정한 주소 함수(FP0, FP1, P7, FP8, FP9)를 선택합니다. FPSEL Register에 DISABLE를 지정하면, 주소는 선택되지 않기 때문에 본 명령을 기술하지 않을 때와 같게 됩니다.

FPSEL Register는, Pattern Program중의 REGISTER문 또는 Test Plan Program의 REG MPAT문에 의해서 지정할 수 있습니다.

REG MPAT문으로 FPSEL Register를 지정하는 것으로써, Test Plan Program으로부터 주소함수의 선택을 할 수 있습니다.

**[Exercise]**

* + **DSD, RCD, CCD Field**

**[Function]**

주소 함수에 의한 Pattern Data의 발생에 사용되는 DSD, RCD, CCD 의 각 Register의 값 을 변경 하는 명령입니다.

**[Format ]**

**DSD<0**

**RCD<0**

**CCD<0**

**DSD<DSD**

**RCD<RCD**

**CCD<CCD**

**DSD<DSD+1**

**RCD<RCD+1**

**CCD<CCD+1**

**RCD<RCD+1 CCD<CCD+1**

**[Description]**

▶DSD<DSD

본 명령은, DSD Register의 값을 Hold하는 명령입니다

▶RCD<RCD

본 명령은, RCD Register의 값을 Hold하는 명령입니다

▶CCD<CCD

본 명령은, CCD Register의 값을 Hold하는 명령입니다

▶DSD<0

본 명령은, DSD Register의 값을 0 (Clear)하는 명령입니다.

▶RCD<0

본 명령은, RCD Register의 값을 0 (Clear)하는 명령입니다.

▶CCD<0

본 명령은, CCD Register의 값을 0 (Clear)하는 명령입니다.

▶DSD<DSD+1

본 명령은, FP3 의 발생에 사용되는 DSD Register의 값을 Increment 하는 명령입니다.

▶RCD<RCD+1

본 명령은, FP8 의 발생에 사용되는 RCD Register의 값을 Increment 하는 명령 입니다.

▶CCD<CCD+1

본 명령은, FP9 의 발생에 사용되는 CCD Register의 값을 Increment 하는 명령 입니다.

▶RCD<RCD+1 CCD<CCD+1

본 명령은, RCD Register와 CCD Register의 값을 동일 Cycle로 Increment하는 명령입니다.

**[Exercise]**

* 1. **Data Inversion Field**

**[Function]**

TP, TP2의 Register의 Data를 반전 합니다.

**[Format ]**

/D

/D2

**[Description]**

▶ /D

TP Register의 Data를 반전합니다. 반전 Bit는 DCMR Register에서 지정합니다.

(DCMR Register의 “0”Bit에 대응하는 TP Register가 반전합니다. DCMR의 지정이 없는 경우 DCMR=#0이 됩니다.)

▶ /D2

TP2 Register의 Data를 반전합니다. 반전 Bit는 DCMR2 Register에서 지정합니다.

(DCMR2 Register의 “0”Bit에 대응하는 TP2 Register가 반전합니다. DCMR의 지정이 없는 경우 DCMR=#0이 됩니다.)

**[Exercise]**

REGISTER

DCMR = #FFF000

:

START #0

; DA0-24

NOP TP<0 ; X

NOP ; #0

NOP /D ; #FF DA12-23은 반전되지 않습니다.

* 1. **Data Flag에 의한 반전(DFLG)**

**[Function]**

TP Register의 내용을 반전합니다.

**[Description]**

DFLG = 1 때 TP Register의 내용을 반전합니다.

DFLG Register는, JZD 명령에 의해서 반전할 수 있습니다.

【 메모 】

DFLG에 의해 TP2 레지스터의 내용을 반전하는 경우, MODE문으로 TP2INV를 지정해 주세요.

**[Exercise]**

START #0

NOP XB<0 YB<0 TP<0

LB0:JNI1 LB0 W XB<XB+1 YB<YB+1^BX X<XB Y<YB

LB1:JNI1 LB1 R XB<XB+1 YB<YB+1^BX X<XB Y<YB

JZD LB0

STPS

이 Step이 2회 실행됩니다.

1회째는, DFLG=0

2번째는, DFLG=1으로 TP레지스터의 내용을 반전합니다.

* + **Data Hold Field**

**[Function]**

이전 cycle의 Data를 그대로 출력합니다

**[Format]**

**PD**

Data Hold 명령어 입니다.

**[Description]**

PD를 기술한 Cycle에서는, DA0-23, DB0-23에는 전의 Cycle의 DA0-23, DB0-23이 그대로 출력됩니다.

**[Exercise]**

EX1. PD기술된 Line의 Data 변화

;TP0-47 DA0-23

NOP TP<0 ; X X

NOP TP</TP ; #0 #0

NOP TP<0 ; #FFFFFFFFFFFF #FFFFFF

NOP PD ; #0 #FFFFFF

NOP PD ; #0 #FFFFFF

NOP ; #0 #0

* + **Source Data Field**

**[Function]**

Source data field 명령은, PDS(Pin Data Select)에 출력하는 Data를 Real-time으로 선택하는 명령입니다.

**[Format ]**

**D<TP1**

**D<TP2**

**D<TPXOR**

**[Description]**

Pattern Program의 DATA DEFINE문, 또는 Test Plan Program중에 SELECT PDS DATA문장으로, Test Data에 D를 지정하면, Pin문의 D0 ~ 23에 DA0 ~ 23의 Data를 출력해, SD0 ~ 23에 DB0 ~ 23의 Data를 출력합니다.

이 때, Source data field 명령과 MODE TPMn의 지정에 의해, DA0 ~23, DB0 ~ 23에 출력하는 Data는 아래 표와 같이 선택됩니다.

Source data field 명령을 생략 하면, D<TP1를 지정했을 경우와 같게 됩니다.

★Test Data 에 D를 지정했을 경우의 DA0 ~ 23, DB0 ~ 23에의 출력 Data

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| MODE TPM24 | |  | | OUTPUT DATA | |
| 48……………………………………0 | |  | |
| 24……DB….. 0 | 24……DA….. 0 |  |  |
| 24……TP2…..0 | 24……TP1..…0 |  | | D<TP1 (TP1 data, Default) | |
| 24……TP2…..0 | 24……TP2…..0 |  | | D<TP2 (TP2 data) | |
| 24……TP2…..0 | 24…TPXOR…0 |  | | BYTEMASK  DEFINE문  지정안함 | D<TPXOR  (TP XOR data) |
| 48…………TPXOR.……………….0 | |  | | BYTEMASK  DEFINE문  지정안함 |

TPXOR는, TP1.XOR.TP2의 연산 결과를 의미합니다.

* + **Program 작성상 주의**

▶Data Bit의 반전 조건의 중복

Data Bit DA0-23, DB0-23의 반전 조건이 중복 했을 경우, 반전 조건이 홀수 회 실행할 때만 Data가 반전되어 Data Bit에 출력됩니다. 반전 조건이 짝수회 실행 되었을 때는 Data의 반전은 행해지지 않습니다.

Data의 반전 조건에는 이하의 것들이 있습니다.

/D, /D2, DFLG, FP함수, ARIRAM(영역 반전)

▶Data 연산의 Default

TP1<TP1, TP2<TP2, D<TP1

* **Control Bit**
  + **Control 명령의 구성**

▶Control 부는 아래와 같은 각 명령으로 구성되어 있습니다.

|  |  |
| --- | --- |
| **Field Name** | **Control bit** |
| **TS field** | TS1 ~ TS32 |
| **MUT field** | R、W、W2、R2、C0-31 |
| **Address control field** | XT, YT |
| **SCRAM control field** | SCROFF, DUT SCROFF, FM SCROFF |
| **PRESCRAM control field** | PSCROFF |
| **ARIRAM control field** | ARIOFF |
| **Pin Data Select field** | PDS1-PDS32、PDS1A- PDS32A、PDS1B- PDS32B |
| **X Address save field** | XASV, XALD |
| **Y Address save field** | YASV, YALD, YASL |

* + **TS field**

**[Function]**

DUT 에 인가하는 주소 및 Data, Clock등을 발생하기 위한 Clock의 주기와 위상은,

Pattern Program중의 이 Control bit 명령에 의해서 Cycle 마다 변경할 수 있습니다.

**[Format ]**

**TSn n=1~32**

Default : TS1 (기술하지 않는 경우)

**[Description ]**

**[Exercise]**

|  |
| --- |
| PRO program-name  :  RATE(1, 100ns, 200ns, 300ns, 400ns);  BCLK(x0, 1, 50ns, 100ns, 150ns, 200ns);  :  END |

|  |
| --- |
| MPAT program-name  :  NOP TS1  NOP TS2  NOP TS3  NOP TS4  :  END |

* + **MUT field**

**[Function]**

R、W、C0-31은, 이하에 나타내는 용도로 사용됩니다.

**[Format ]**

R、W、C0-31

**[Description]**

DUT의 WE, CS, OE의 제어 단자의 Data Pattern으로서 사용합니다.

논리 비교의 Enable/Disable의 제어에 사용합니다.(CPE1 ~ 4)

I/O pin의 Driver의 Driver Enable/Disable의 제어에 사용합니다.(DRE1 ~ 2)

Real-time Hi-Z Mode로 Hi-Z비교하는 Cycle을 지정할 경우에 사용합니다.

Auto Scan mode로, STRB 및 Clock을 제어하는 Cycle을 지정할 경우에 사용합니다.

**[Exercise]**

* + **Address control field**

**[Function]**

X,Y주소에 XT, YT Register 값을 일시적으로 출력합니다.

**[Format ]**

**XT**

X주소에 XT Register의 값을 일시적으로 출력합니다.

**YT**

Y주소에 YT Register의 값을 일시적으로 출력합니다.

**[Description]**

주소 Line으로부터 Command 설정을 실시하는 DUT의 측정에 있어서, 주소 Line에 대해서 일시적으로 논리 Data를 출력할 때 등에 사용합니다

**[Exercise]**

* + **SCRAM control field**

**[Function]**

주소의 Descramble을 일시적으로 오프 합니다.

**[Format ]**

**SCROFF**  (DUT SCROFF와 FM SCROFF를 동시에 지정)

**DUT SCROFF** (DUT 로의 Descramble을 일시적으로 OFF)

**FM SCROFF**  (Fail Memory로의 Descramble을 일시적으로 OFF)

**[Description]**

1. Address Descramble Memory를 사용하고, ALPG 로 발생한 주소를 변환하고 있는 경우, 일시적으로 다음의 변환을 그만두고 싶을 때에 사용합니다.

2. XT, YT와 조합하고, 주소 Line으로부터 Command 설정을 실시할 때 등에 사용합니다

**[Exercise]**

* + **PRESCRAM control field**

**[Function]**

PRE-SCRAM부에 의한 Address 변환 동작을 일시적으로 OFF 합니다.

**[Format ]**

**PSCROFF**

**[Description]**

**[Exercise]**

* + **ARIRAM control field**

**[Function]**

영역 반전 Memory에 의한 Data의 반전을 일시적으로 OFF 합니다.

**[Format ]**

**ARIOFF**

**[Description]**

**[Exercise]**

* + **Cycle Palette field**

**[Function]**

Cycle 마다 ALPG 의 출력 Data와 Pin Data의 할당을 Real time로 전환하는 명령입니다.

**[Format ]**

**CYPAn(CYPn)** n=1~8

**CYPAn** n=1~8

**[Description]**

Pattern Program중의 CYPAn, CYPBm의 지정에 의해서, Cycle 마다 Pin Data를 바꾸는 일이 생깁니다. (CYPAn, CYPBm의 지정은, 지정하는 Cycle에서만 유효합니다.)

CYPAn, CYPBm와 Pin Data의 대응은, Pin문의 Data Selector와 CYPA, CYPB로 지정합니다.

【 메모 】

►CYPAn는 CYPn와 기술할 수 있습니다.

►CYPAn, CYPBm의 기술을 생략 한 Cycle에는, CYPA1, CYPB1를 할당할 수 있습니다.

►Pin문으로 Cycle palette의 Pin Data를 생략 했을 경우, 이전 지정한 Data가 보관 유지됩니다.

【 제한 】

►CYPAn는 동일 Step에 복수 기술할 수 없습니다.

►CYPBm는 동일 Step에 복수 기술할 수 없습니다.

►CYPAn와 CYPBm는 동일 Step에 기술할 수 있습니다.

**[Exercise]**

:

P23=...,CYPA, CYP1,<D0>, <D1>, <D2>, <D3> ;

P25=...,CYPB, CYP1, <SD0>, <SD1>, <SD2>,<SD3>;

MPAT ~~program-name~~

:

NOP CYPA1 CYPB2 ; P23에 D0, P25에 SD1

NOP CYPA2 CYPB3 ; P23에 D1, P25에 SD2

NOP CYPA3 CYPB4 ; P23에 D2, P25에 SD3

NOP CYPA4 CYPB1 ; P23에 D3, P25에 SD0

:

END

* + **X Address save field**

**[Function]**

복수의 Bank를 가지는 Memory Device에서는, 각 Bank를 독립해 제어(Multi Bank Operation) 할 수 있습니다.

**[Format ]**

XASV

XALD

**[Description]**

▶XASV

1. XASV를 지정한 Cycle의 Row Address(ALPG로부터의 X주소)를 Save합니다.

2. Bank에 대해서 Active Command를 입력해 Row Address를 주는 Cycle로 지정합니다.

3.BANK SAVE ADDRESS문으로 Row측의 Bank 주소를 지정하는 것으로, Bank 마다 Row Address를 Save할 수 있습니다. (최대 64개의 Row Address를 Save할 수 있습니다.)

▶XALD

1. XASV로 Save한 Row Address(ALPG로부터의 X주소)를, XALD를 지정한 Cycle로 Load합니다.

2. Bank에 대해서 Column 주소를 주어 Read Command를 입력하는 Cycle로부터 Data Read Cycle mode로 지정합니다.

3. BANK LOAD ADDRESS문으로 Column측의 Bank 주소를 지정하는 것으로, Bank 마다 Save 한 Row Address를 Load 옵니다.

【 메모 】

XASV와 XALD는 동일 Step에 기술할 수 있습니다. 이 경우, XASV를 실행 후, XALD를 실행합니다.

XALD 명령으로 Load하는 X주소는, SELECT PRESCRAM ADDRESS문의 지정에 따라 출력됩니다.

**[Exercise]**

* + **Y Address save field**

**[Function]**

복수의 Bank를 가지는 Memory Device에서는, 각 Bank를 독립해 제어(Multi Bank Operation) 할 수 있습니다.

**[Format ]**

**YASV**

**YALD**

**[Description]**

▶YASV

YASV를 지정한 Cycle의 Row Address(ALPG로부터의 Y주소)를 Save합니다.

Bank에 대해서 Active Command를 입력해 Row Address를 주는 Cycle로 지정합니다.

BANK SAVE ADDRESS문으로 Row측의 Bank 주소를 지정하는 것으로, Bank 마다 Row Address를 Save할 수 있습니다. (최대 64개의 Row Address를 Save할 수 있습니다.)

▶YALD

YASV로 Save한 Row Address(ALPG로부터의 Y주소)를, YALD를 지정한 Cycle로 Load합니다.

Bank에 대해서 Column 주소를 주어 Read Command를 입력하는 Cycle로부터 Data Read Cycle mode로 지정합니다.

BANK LOAD ADDRESS문으로 Column측의 Bank 주소를 지정하는 것으로, Bank 마다 Save 한 Row Address를 Load 옵니다.

▶YASL

YASL 명령을 지정한 Cycle에서는, YASV 명령으로 FIFO Memory에 저장한 column 주소를, YALD 명령으로 지정한 Read address Pointer로부터 읽어내, FM, FP함수, ARIRAM에 출력합니다. Read Comment입력한 Cycle의 CAS 지연시간 후의 Cycle로부터 Read Cycle 모두에 지정합니다.

【 메모 】

YASV와 YALD는 동일 Step에 기술할 수 있습니다. 이 경우, YASV를 실행 후, YALD를 실행합니다.

YALD 명령으로 Load하는 Y주소는, SELECT PRESCRAM ADDRESS문의 지정에 따라 출력됩니다.

**[Exercise]**