Control Flow Graph Generator

(SASD Supplement)

- T11 SASD 수정 -

T10

200811436 안정무

200811437 여종훈

200811406 권성광

200811460 최산수(v)

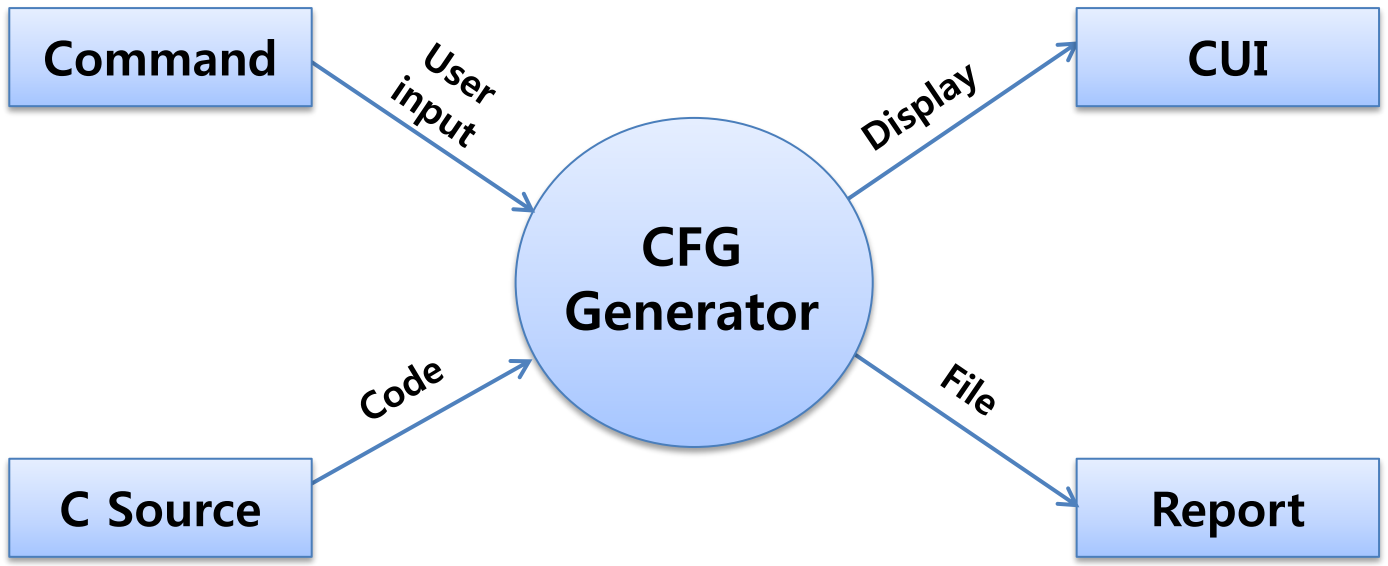
Statement of Purpose

1. Input are the file written by C-language and commands are entered by the user.
2. The output are CUI and report File that holds the information of the CFG.
3. Block C Source files received as input to the divide, this is a program that generates a CFG to Block..
4. Execution of the software is in the form of a Command Line Commands.
5. When you have entered the wrong type of command to output to help.
6. 100 to 200 lines of code size, the program is targeted, code should include Main Function.
7. The code works for a single file. With a user-defined header file is not guaranteed to work for.
8. Do not use the pointer to the code is targeted at.

Statement of Purpose(Cont')

9. Every statement is specified using a number, order of the files that received input are same as Numerical order.
10. Need to process about function calls, for statements, while statements, switch statement
11. Between Block and Block is connected to the appropriate edge
12. CFG is composed of the result of the conversion Report, Block, Edge, Error Message.

System Context Diagram



Event List

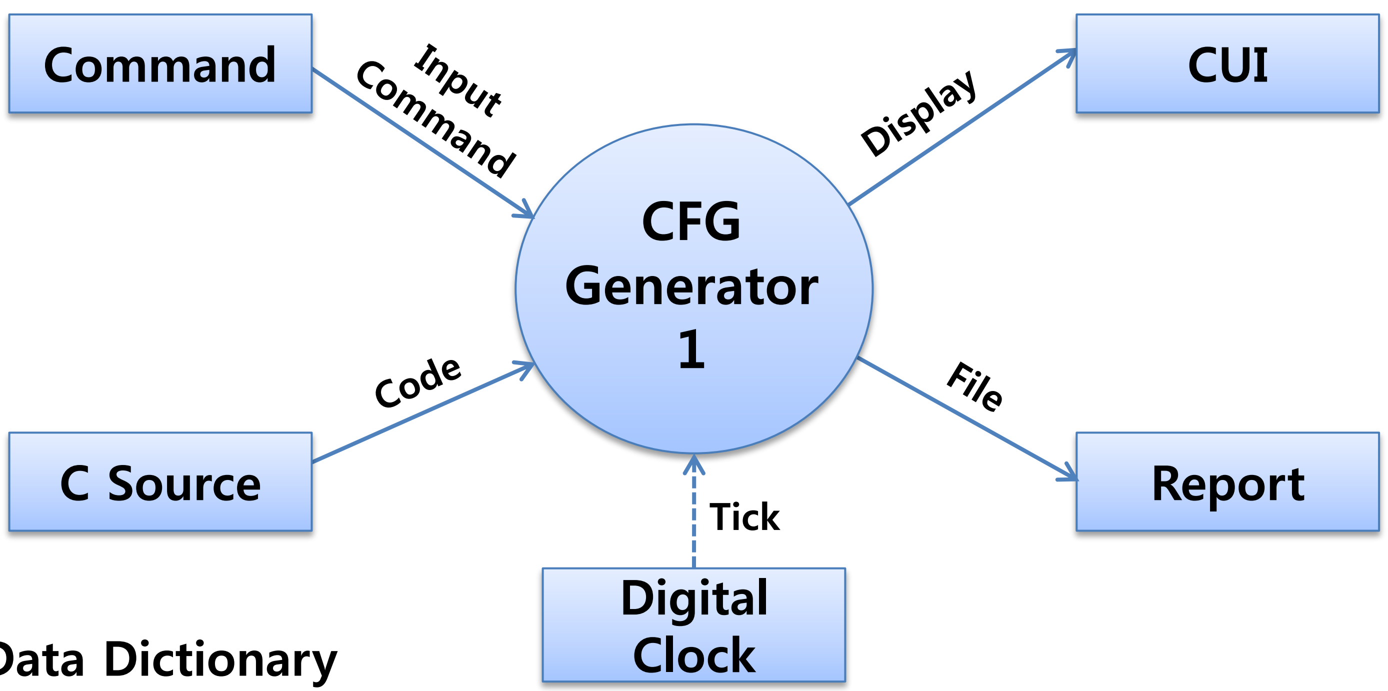
◆ Input Event

Input Event	Description	Format / Type
Code	Successfully of around 100 to 200 lines C Source to work	*.C file
User Input	command of input ex) #gcc ./CG Inputcode.c result.txt	gcc Commands

◆ Output Event

Output Event	Description	Format / Type
Display	The resulting of CFG needs to be output as the CUI	Display
File	the resulting of CFG need to be output in a file	Text file

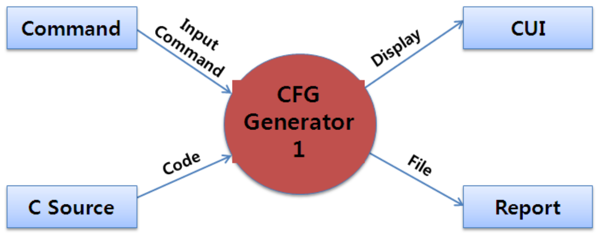
DFD Level 0

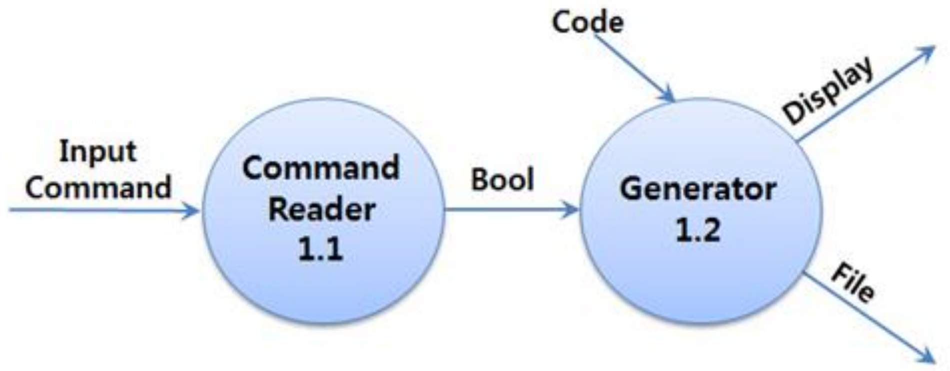


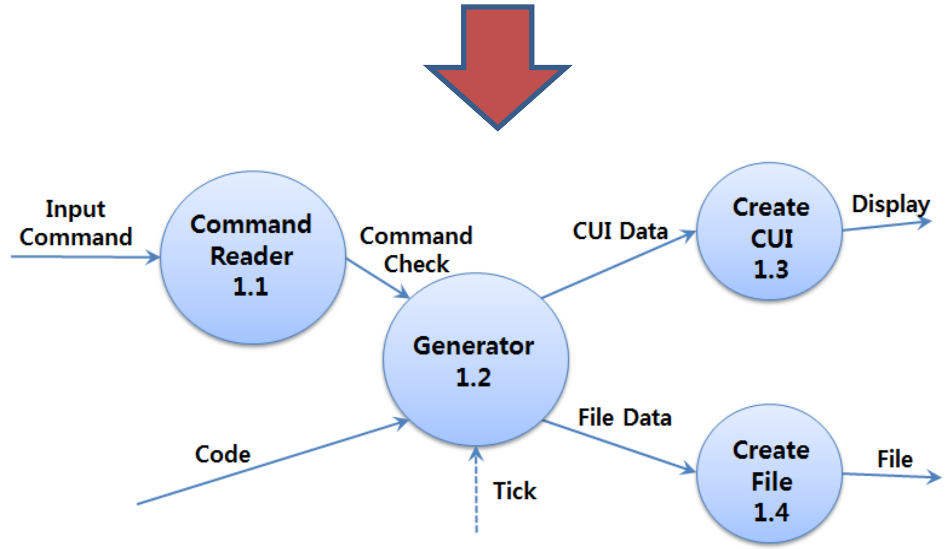
◆ **Data Dictionary**

Data	Description	Format / Type
Input Command	As the user enters a command to the input C Source File of the path to receive and specifies File name to output CFG.	String
Code	C source files to work properly.	*.c
Display	CFG completed output to the console screen.	Console Display
File	Generating the file from completed CFG.	*.txt

DFD Level 1

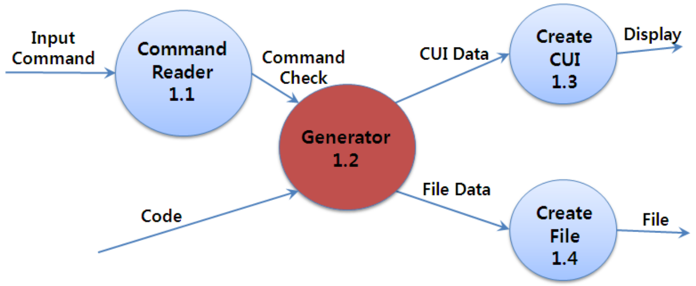


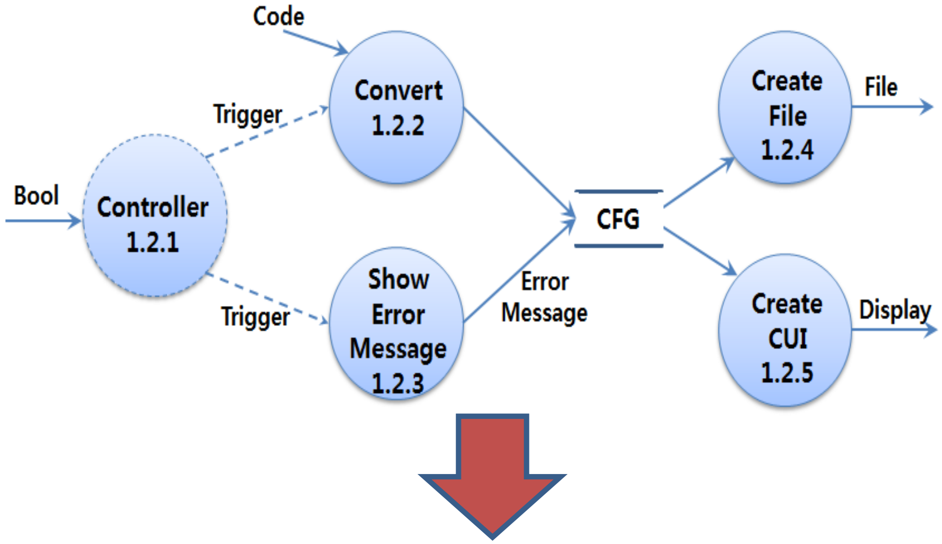


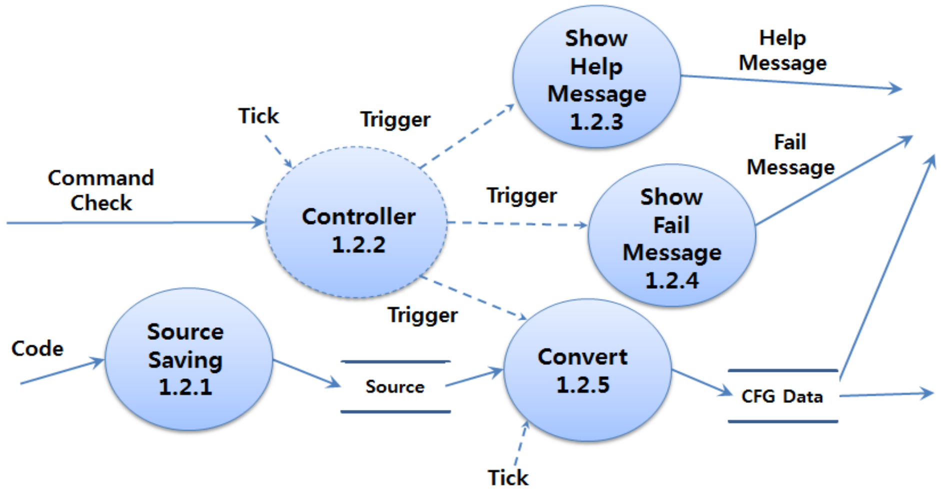


Generator does not correspond to the output process.
so output process has divided from Generator to create Create-CUI and Create-File.

DFD Level 2
- Generator



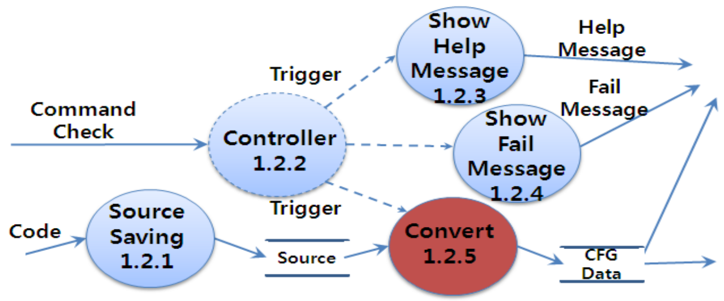


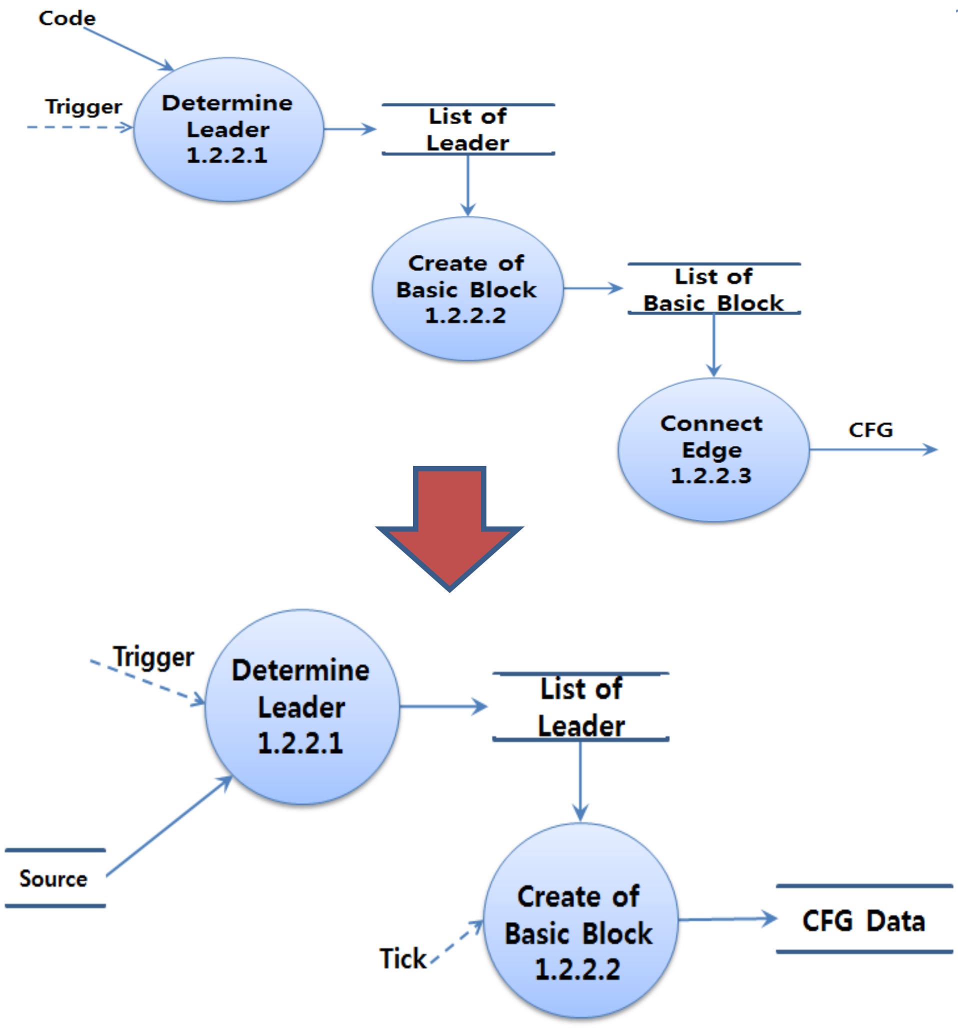


The raw code, it can be difficult to convert. so we added the process for converting a fixed format and saving the code. we changed the data-store to separate error message from CFG-data-store.

DFD Level 3

- Convert

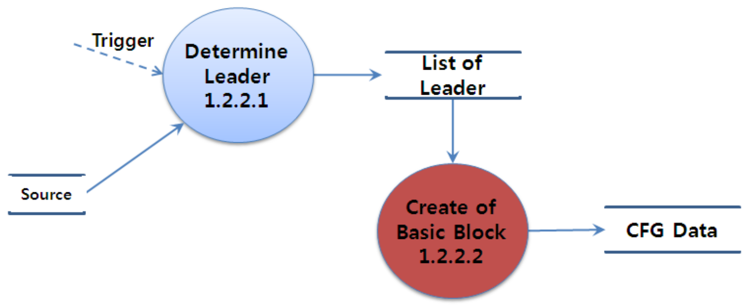


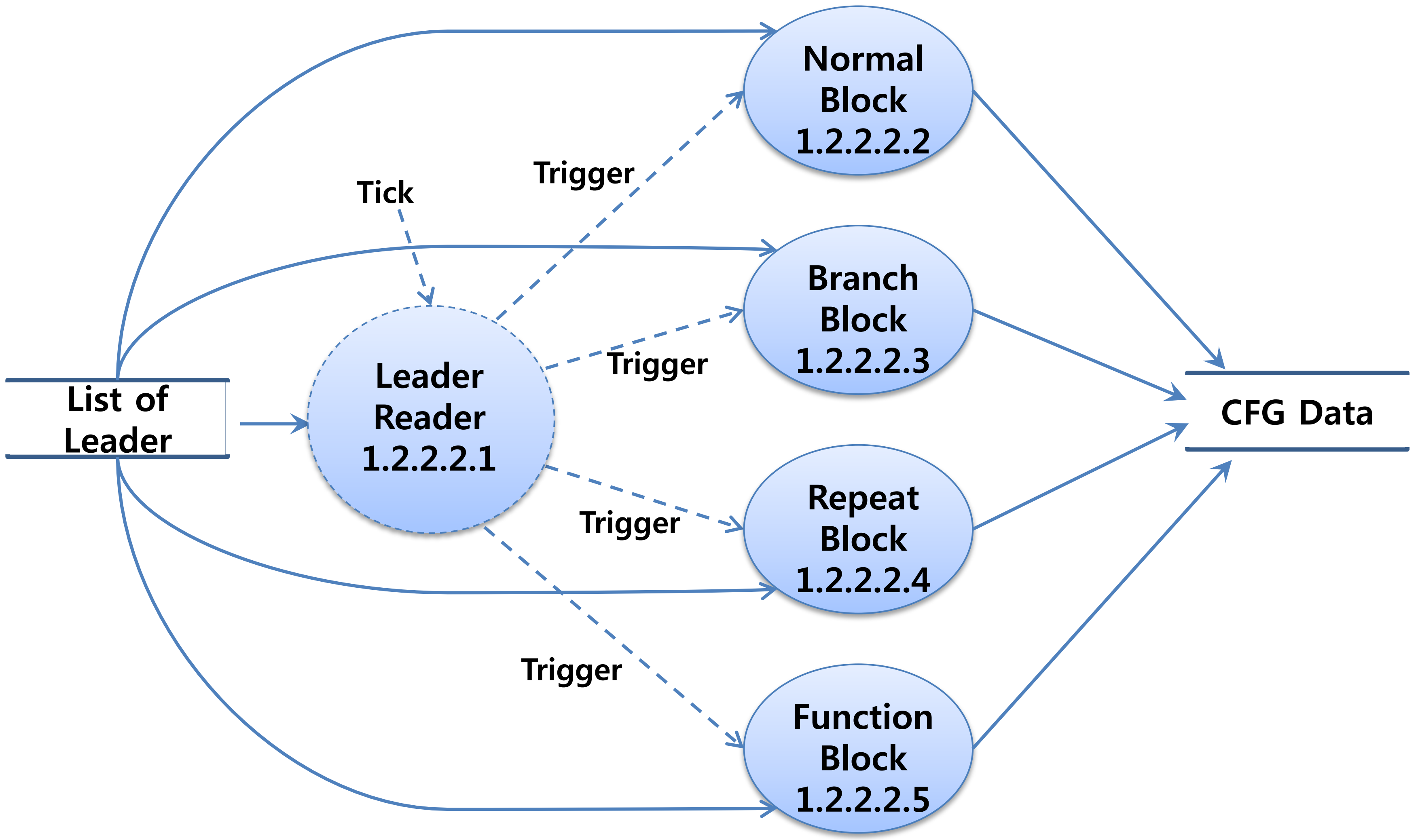


we deleted the Connect-Edge-process because at the same time when processing the block, edge is processed too.
so the role of Connect-Edge is mixed with Create of Basic Block.

DFD Level 4.1

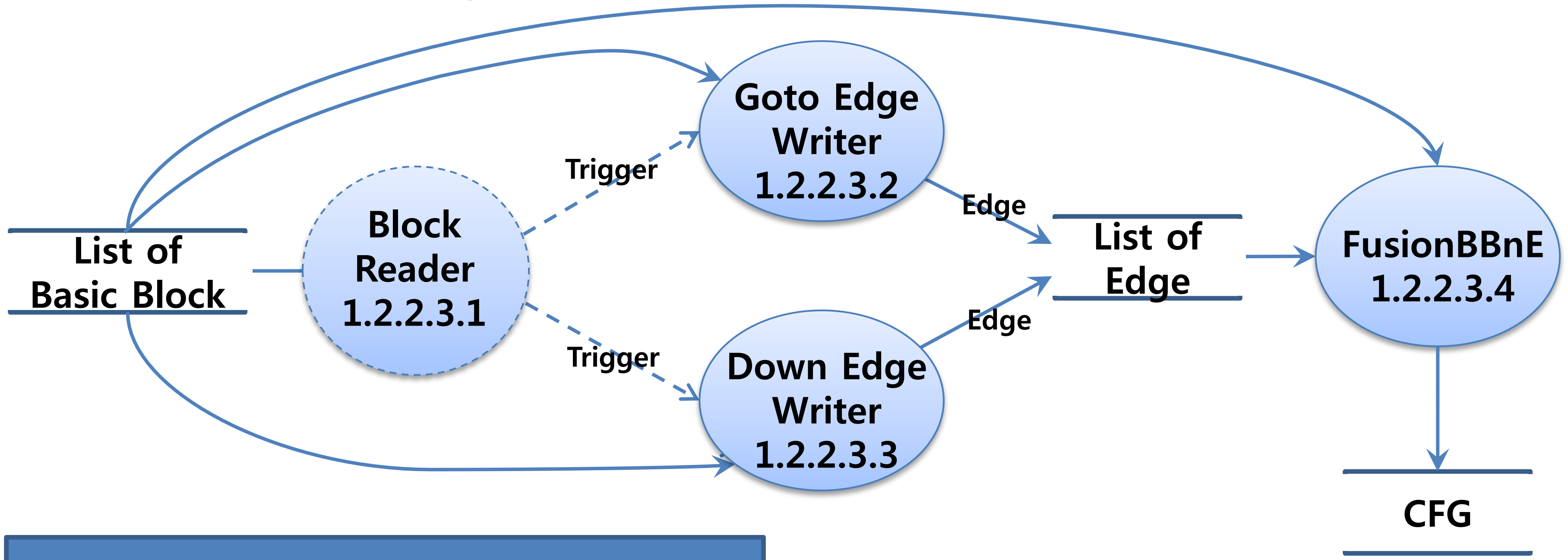
- Create of Basic Block



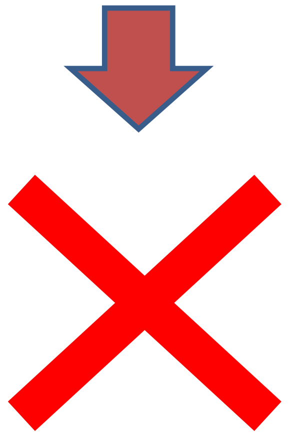


DFD Level 4.2

- Connect Edge

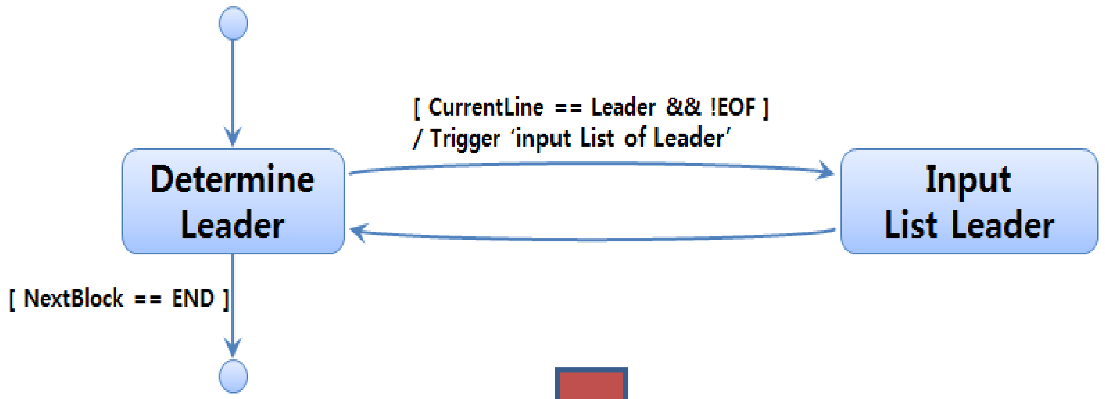


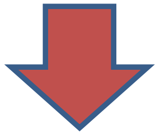
As mentioned above, while processing a block and processing the edges at the same time because so this section were deleted.



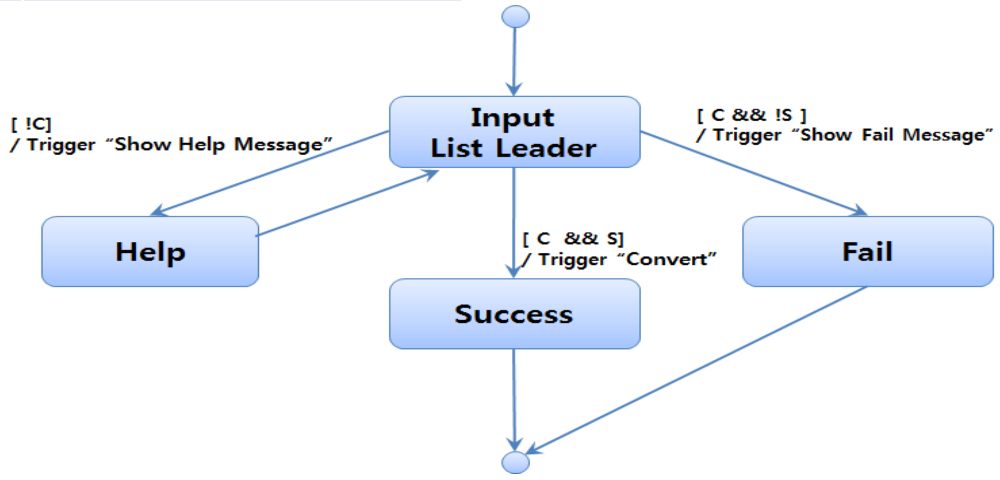
DFD Level 5

- Finite State Machine 1.2.2


[CurrentLine == Leader && !EOF]
/ Trigger 'input List of Leader'
Determine Leader
Input List Leader
[NextBlock == END]



S	Input Source success or not
C	Input Command success or not

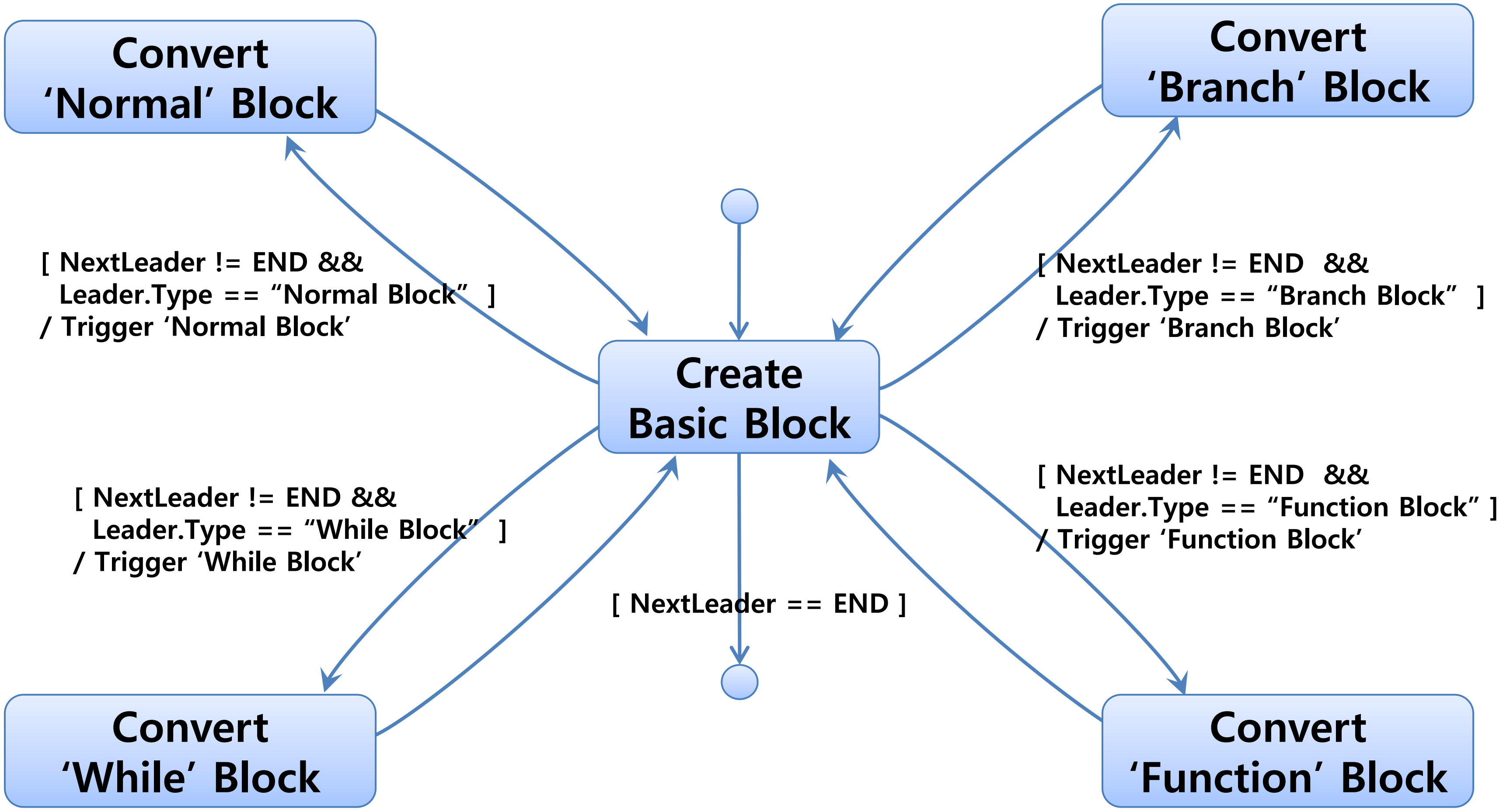


SA was changed to fit the Finite state machine.

DFD Level 5

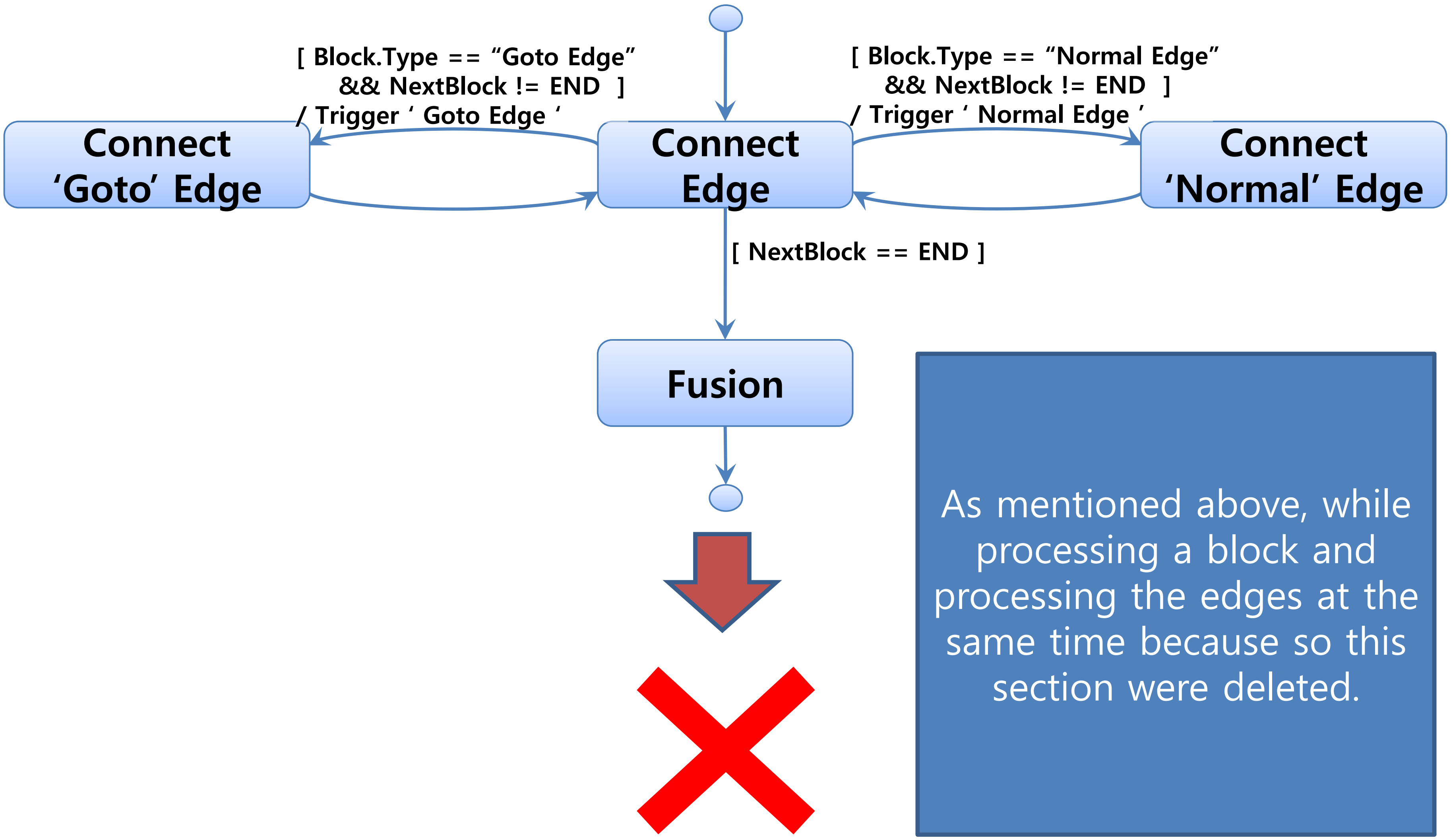
- Finite State Machine 2

NextLeader	Connected leader of next block
Leader.Type	Block's Type


Convert 'Normal' Block
Convert 'Branch' Block
Create Basic Block
Convert 'While' Block
Convert 'Function' Block
[NextLeader != END && Leader.Type == "Normal Block"] / Trigger 'Normal Block'
[NextLeader != END && Leader.Type == "Branch Block"] / Trigger 'Branch Block'
[NextLeader != END && Leader.Type == "While Block"] / Trigger 'While Block'
[NextLeader != END && Leader.Type == "Function Block"] / Trigger 'Function Block'
[NextLeader == END]

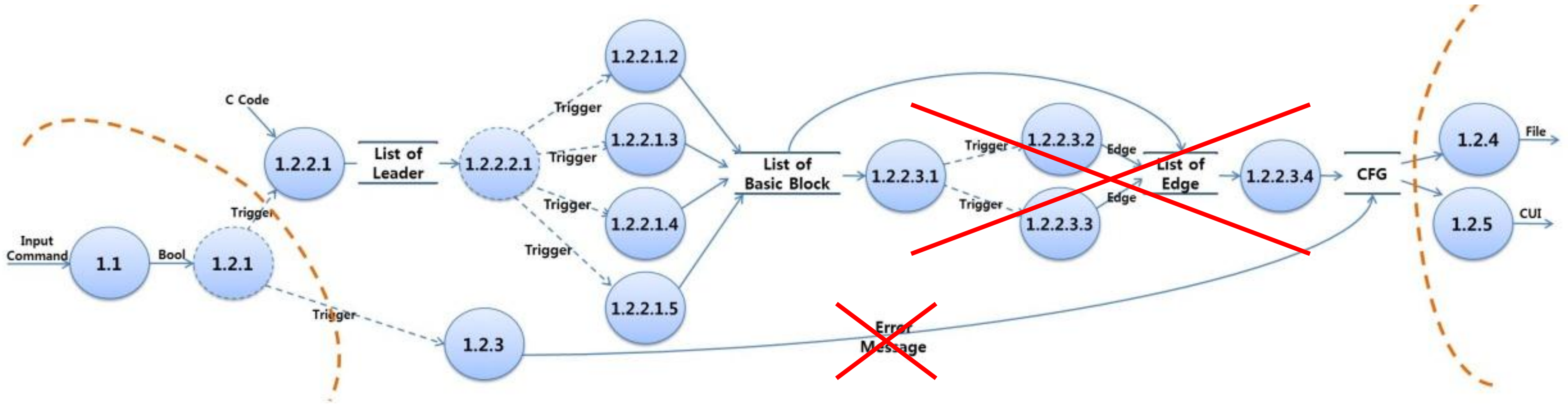
DFD Level 5

- Finite State Machine 3


[Block.Type == "Goto Edge" && NextBlock != END] / Trigger ' Goto Edge '
[Block.Type == "Normal Edge" && NextBlock != END] / Trigger ' Normal Edge '
Connect 'Goto' Edge
Connect Edge
Connect 'Normal' Edge
[NextBlock == END]
Fusion

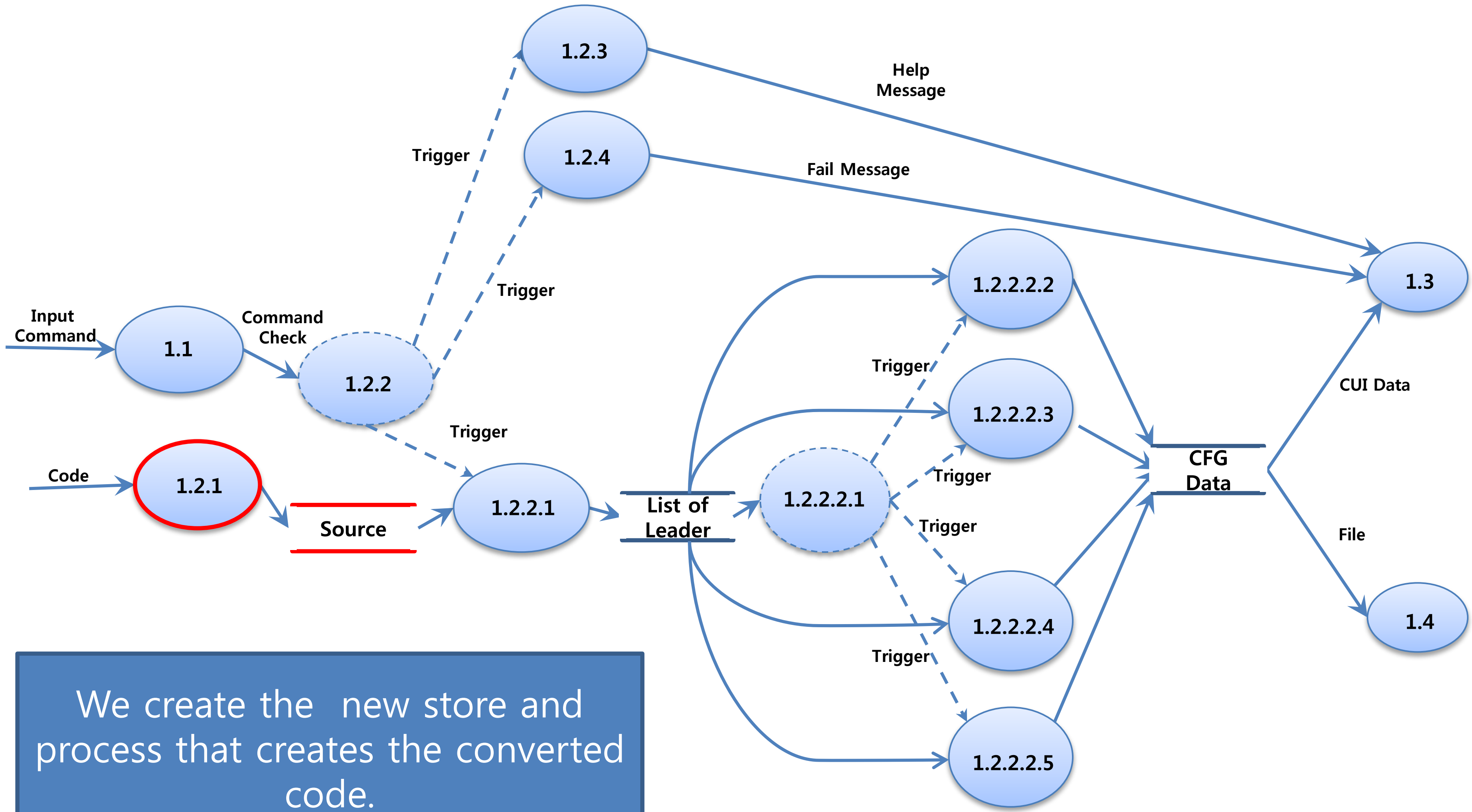
As mentioned above, while processing a block and processing the edges at the same time because so this section were deleted.

DFD – Overall (deleted thing)



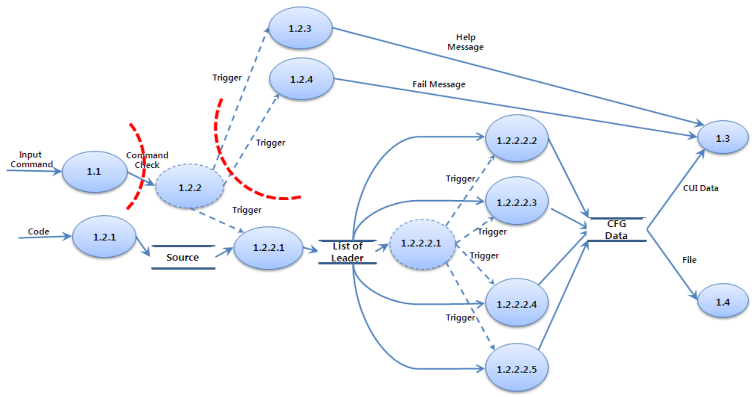
Changing the structure to handle the error message.
The portion of handling the edge has been deleted.

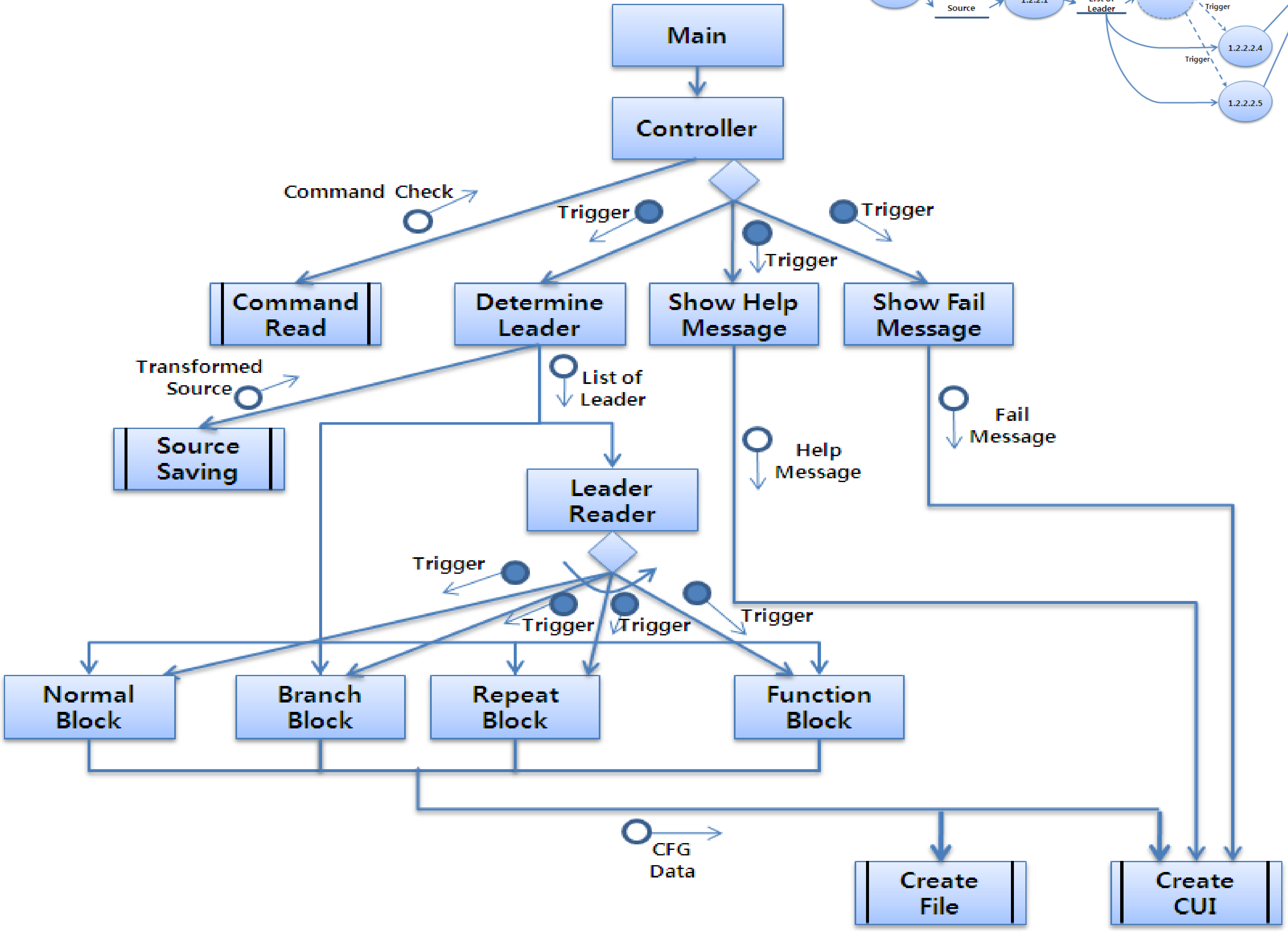
DFD – Overall (새로운 것)



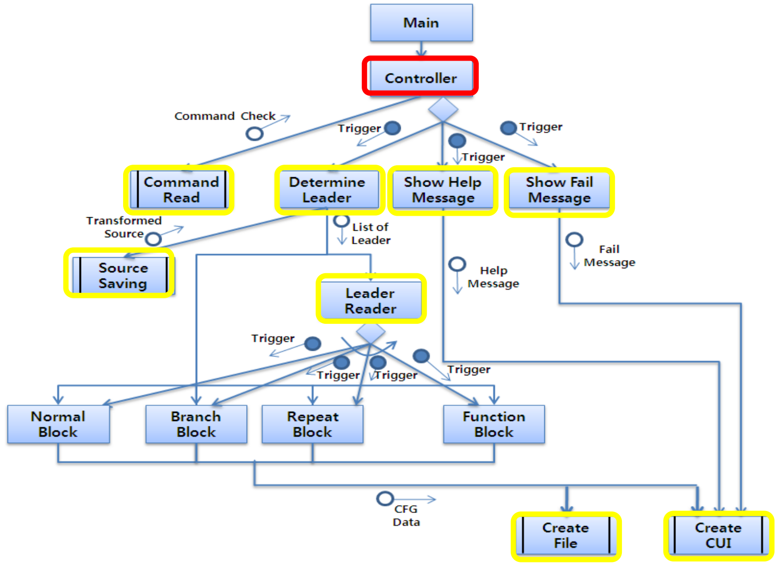
We create the new store and process that creates the converted code.

Structured Chart(Advanced)





Structured Chart(Advanced)



Controller

```
#include"CFG_h.h"

void main( int argc , char **argv )
{
    int i;
    int listcount;
    int Input_Check;
    char Tcode[CSIZE][CSIZE];
    BlockList *BasicList[CSIZE];
    FILE *out = fopen("report.txt","w");

    Input_Check = Generator_source( argv[0] );
    if( Input_Check == -1 )  // C Code를 Transformed Code로 변환
        Print_Help();  // 도움말 출력
    else if( Input_Check == 0 )
    {
        printf("END Program₩n");
        return;
    }
    else
        printf("Input file open success₩n");

    listcount = Determine_Leader( BasicList , Tcode );  // Basic List를 만들어 줌

    for( i = 0 ; i < listcount ; i ++ )  // 리스트의 개수만큼 CFG 생성
    {
        Trans_Block( BasicList[i] , Tcode );  // Basic List를 순회하며 CFG 생성
        Print_CUI_FILE( BasicList[i]->B , Tcode , out );  // 생성된 CFG 데이터를 화면과 파일로 출력
    }

    fclose( out );
}
```

Command Read

Source Saving

Show Help Message

Show Fail Message

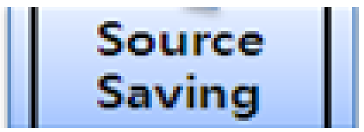
Determine Leader

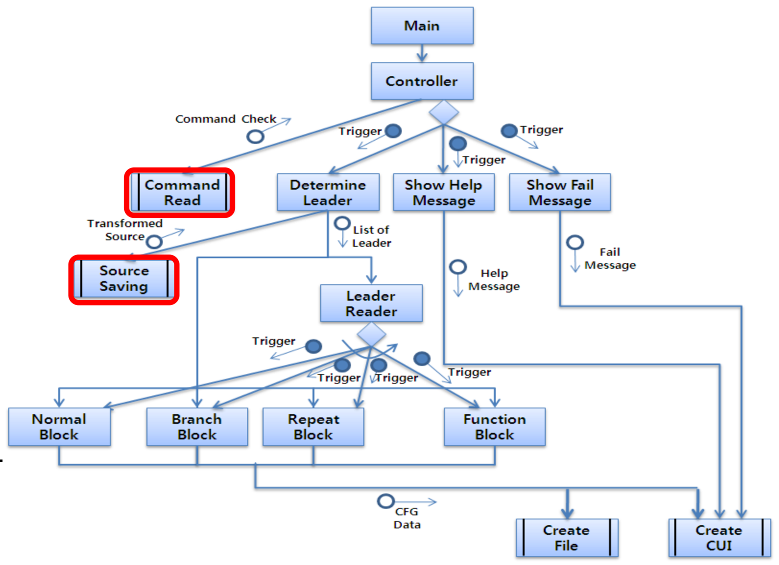
Leader Reader

Create File

Create CUI

Structured Chart(Advanced)

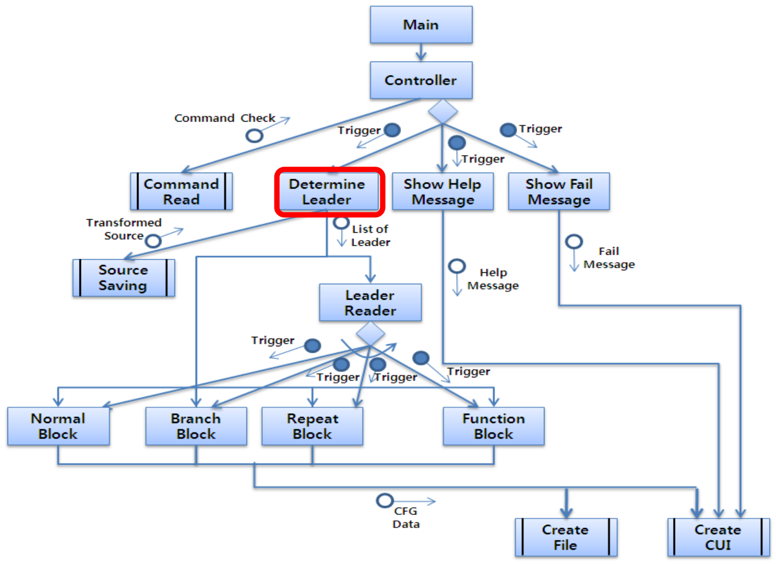




```
int Generator_source(char file_name[])
{
    int i, file_line;
    char **c_source = (char**)malloc(sizeof(char**)*DEFAULT
    memset(c_source, 0x0, sizeof(char**)*DEFAULT_LINE);
    for(i = 0; i < DEFAULT_LINE; i++){
        c_source[i] = (char*)malloc(sizeof(char*)*DEFAULT_CHAR);
        memset(c_source[i], 0x0, sizeof(char*)*DEFAULT_CHAR);
    }

    if( file_name == NULL )
        return -1;

    file_line = Read_txt(c_source, file_name); // 파일읽기
    if(file_line == -1)
        return 0;
    Convert_txt(c_source, &file_line); // 중간변환
    Print_txt(c_source, file_line); // 파일쓰기
    file_line = Read_txt(c_source , "Tcode.txt"); // 다시 읽어서
    // 중간에 의미없는 개행 없애기
    Print_txt(c_source, file_line); //파일쓰면 중간변환 완성
    //file_line 1줄이는 이유는 맨 마지막 } 때문에 line++ 시켜서
    //쓰레기값도 찍힌다.

    free(c_source);
    return file_line;
}
```

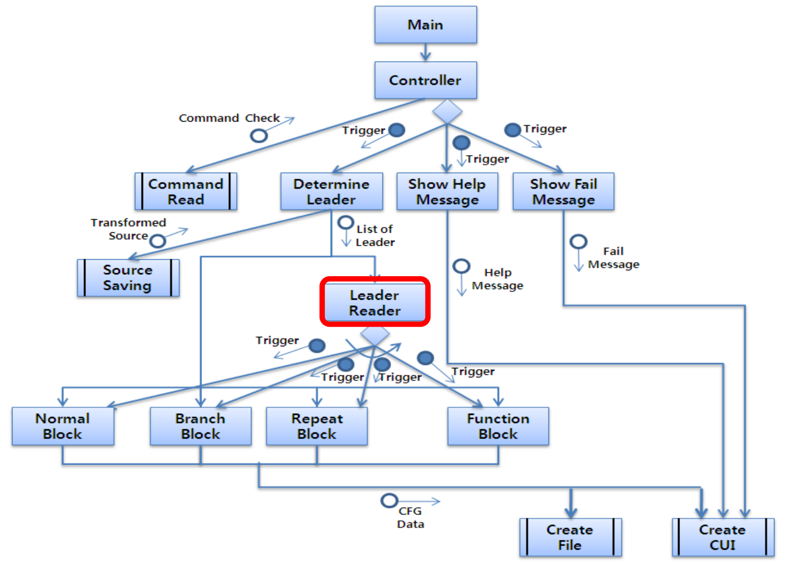
Using Data structure
Stack

Structured Chart(Advanced)



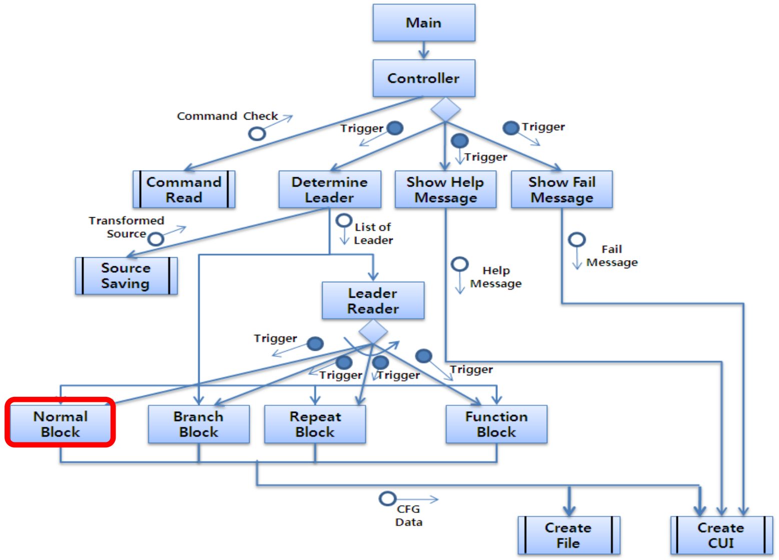
```
int Determine_Leader( BlockList *BasicList[] , char Tcode[][CSIZE] )
{
    int i , EOFline;
    int startnum , endnum;
    int BasicCnt = 0;
    BlockList *now , *next;
    FILE *fin = fopen( "Tcode.txt" , "r" ); // 변환된 코드 파일을 오픈.

    for( i = 0 ; fgets( Tcode[i] , 100 , fin ) != NULL ; i ++ );  // 변환된 코드를 char 배열에 읽어온다.

    EOFline = i;  // 파일의 끝 라인
    |
```

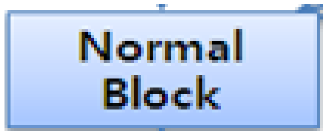
Structured Chart(Advanced)



```
void Trans_Block( BlockList *BasicList , char Tcode[][CSIZE] ) // 블록리스트를 탐색하며 CFG를 완성시켜나간다.
{
    int i , j;
    BlockList *now , *next , *prenext , *endlist;
    int startnum , endnum;

    now = BasicList;
    while( now->nextBL != NULL )  // 블록리스트의 마지막 블록을 찾는다.
        now = now->nextBL;

    endlist = now;

    now = BasicList->nextBL;
```

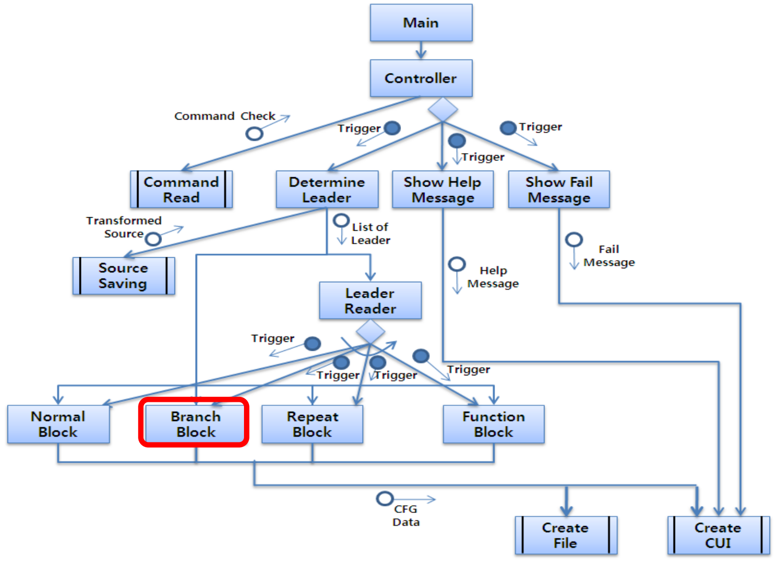
Structured Chart(Advanced)





```
if( now->B->Btype == NORMAL || now->B->Btype == CASE )
{
    endlist = Trans_NormalBlock( now , Tcode , endlist );
    // break 처리
}
```

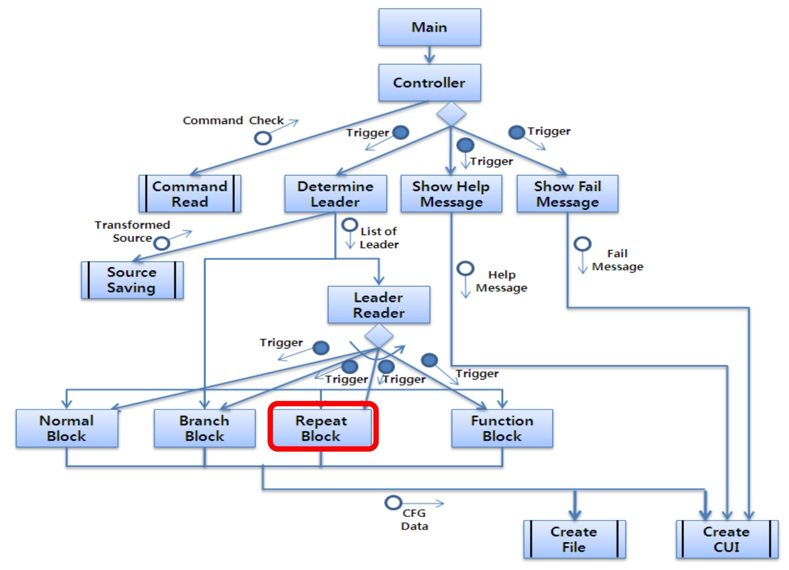
Structured Chart(Advanced)



Branch Block

```
else if( now->B->Btype == BRANCH )
{
    int elsech = 0;   // else 인지 체크하는 변수
    i = now->B->startline;
    if( strncmp( Tcode[i] , "if:" , 3 ) == 0 )  // if 블록 처리
        sscanf( &Tcode[i][3] , "%d" , &startnum );
    else if( strncmp( Tcode[i] , "elseif:" , 7 ) == 0 )  // elseif 블록 처리
        sscanf( &Tcode[i][7] , "%d" , &startnum );
    else if( strncmp( Tcode[i] , "else:" , 5 ) == 0 )  // else 블록 처리
    {
        sscanf( &Tcode[i][5] , "%d" , &startnum );
        elsech = 1;
    }

    for( ; ; i ++ )  // BRANCH 한블록 찾기
    {
        if( strncmp( Tcode[i] , "endif:" , 6 ) == 0 )
        {
            sscanf( &Tcode[i][6] , "%d" , &endnum );
            if( startnum == endnum )
                break;
        }
    }

    /// TRUE 일때 블록생성
    next = Create_BlockList( );
    next->B->Btype = NORMAL;
    next->B->startline = now->B->startline + 1;
    next->B->endline = i - 1;
    next->B->brk = now->B->brk;
    Insert_Edge( next->B , now->B->next[0] , NORMAL );  // 새블록의 NORAML엣지 추가
    if( elsech == 1 )  // else 블록인 경우
    {
        now->B->next[0] = next->B;  // 기존 블록 엣지 수정
        now->B->Etype[0] = TRUE;
    }
    else  // else 블록이 아닌 경우
        Insert_Edge( now->B , next->B , TRUE );  // 기존 BRANCH블록과 새블록을 연결하는 TRUE 엣지 추가
    endlist->nextBL = next;  // 리스트에 추가
    endlist = endlist->nextBL;

    /// BRANCH 상태가 끝났는지 검사
    if( i != now->B->endline )  // 더 BRANCH 블록이 있을경우
    {
        /// FALSE 일때 블록생성
        next = Create_BlockList( );
        next->B->Btype = BRANCH;
        next->B->startline = i + 1;
        next->B->endline = now->B->endline;
        next->B->brk = now->B->brk;
        Insert_Edge( next->B , now->B->next[0] , FALSE );  // 새블록의 FALSE엣지 추가
        endlist->nextBL = next;  // 리스트에 추가
        endlist = endlist->nextBL;

        now->B->next[0] = next->B;  // 기존 블록 엣지 수정
    }

    now->B->endline = now->B->startline;   // 기존 블록 endline 수정
}
```

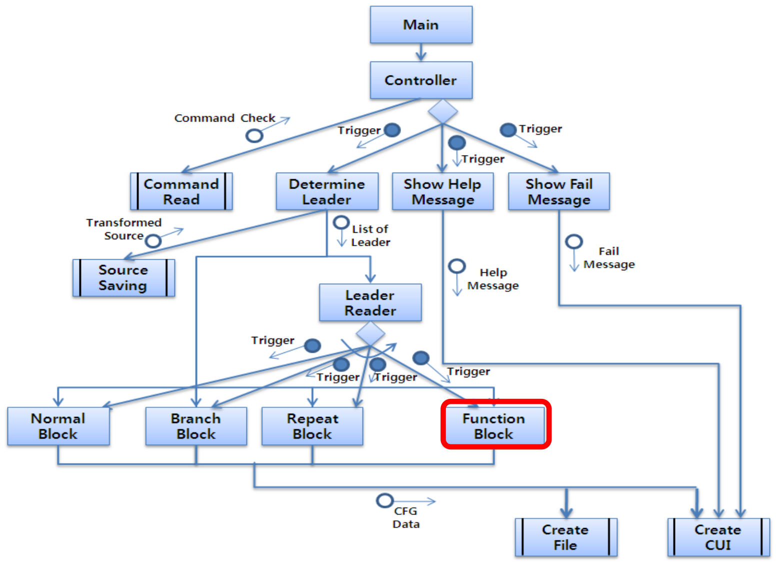
Structured Chart(Advanced)



Repeat Block

```
else if( now->B->Btype == WHILE )
{
    next = Create_BlockList( );
    next->B->Btype = NORMAL;
    next->B->startline = now->B->startline + 1;
    next->B->endline = now->B->endline - 1;
    next->B->brk = now->B->next[0];
    Insert_Edge( next->B , now->B , LOOP );

    now->B->endline = now->B->startline;
    Insert_Edge( now->B , next->B , TRUE );

    endlist->nextBL = next;  // 리스트에 추가
    endlist = endlist->nextBL;
}
```

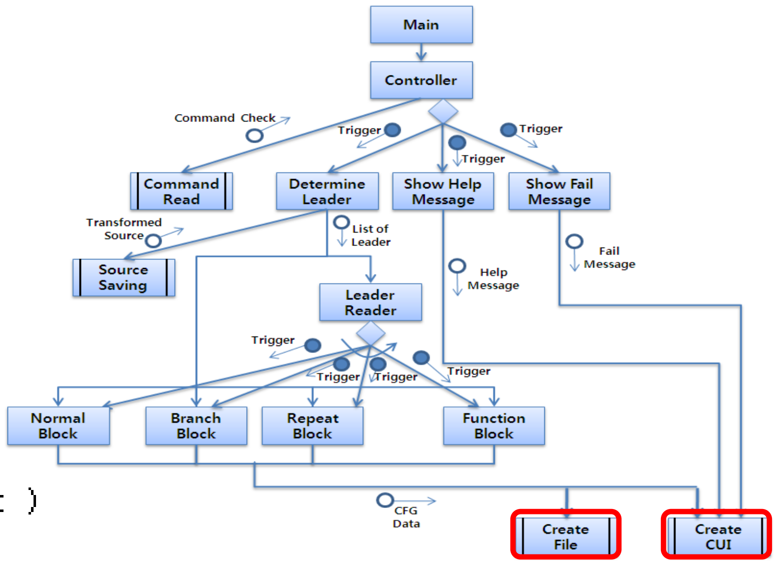
Structured Chart(Advanced)



Function Block

```
if( strncmp( Tcode[i] , "function:" , 9 ) == 0 )  // 함수 별로 CFG 처리, BasicList 헤더 생성
{
    BasicList[BasicCnt] = Create_BlockList();
    now = BasicList[BasicCnt];
    now->B->startline = i;
    now->B->endline = i;
    now->B->Btype = FUNCTION;
}
```

Structured Chart(Advanced)

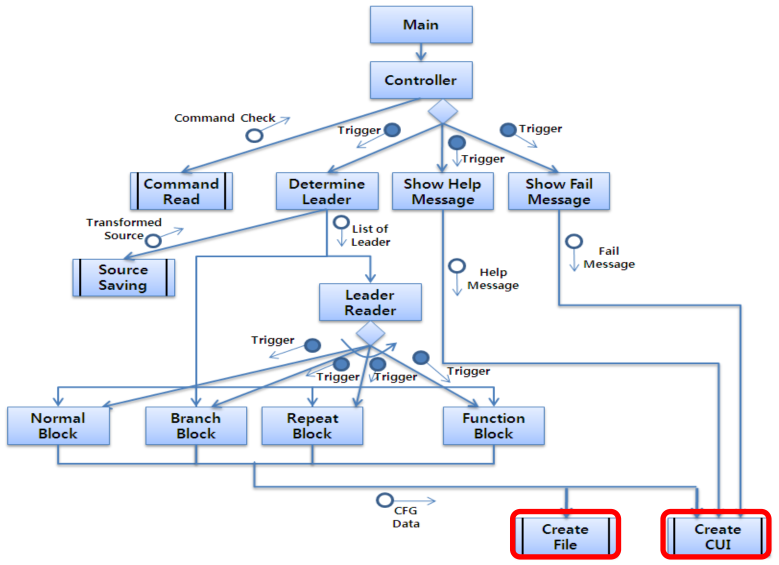


```
void Print_CUI_FILE( Block *B , char Tcode[][CSIZE] , FILE *out )
{
    int i , j;
    Queue Q;
    Block *now;

    Create_Q( &Q );
    Insert_Q( B , &Q );
    B->visit = 1;
     // BFS로 모든 노드 탐색
    while( Q.size )
    {
        now = delete_Q( &Q );

        for( i = 0 ; i < now->nextcnt ; i ++ )
        {
            if( now->next[i]->visit == 0 )
            {
                Insert_Q( now->next[i] , &Q );
                now->next[i]->visit = 1;
            }
        }
    }
```

Using Algorithm BFS

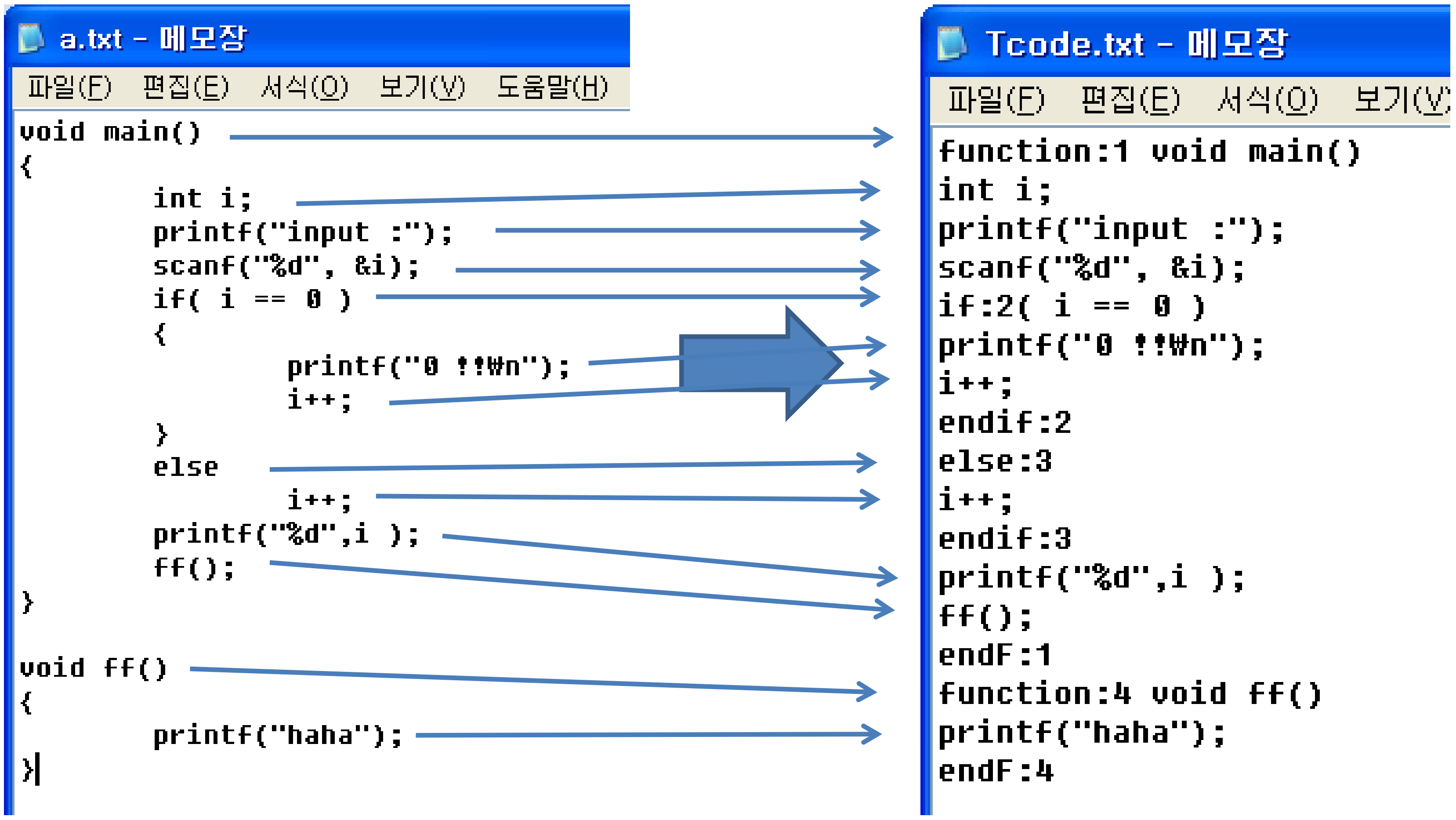
```
 // Block Number를 기준으로 정렬
for( i = 0 ; i <= Q.rear ; i ++ )
{
    for( j = i+1 ; j <= Q.rear ; j ++ )
    {
        if( Q.element[i]->startline > Q.element[j]->startline )
        {
            now = Q.element[i];
            Q.element[i] = Q.element[j];
            Q.element[j] = now;
        }
    }

    now = Q.element[i];
```

Structured Chart(Advanced)



```
// CUI
printf("₩n");
printf("*******************************************₩n");
printf("Block Number : %d₩n" , now->startline ,
Print_BType( now , Tcode );
printf("Connected Edge : ₩n(Edge type-Block Number)₩n");

// FILE
fprintf(out,"₩n");
fprintf(out,"*******************************************₩n");
fprintf(out,"Block Number : %d₩n" , now->startline ,
FPrint_BType( now , out , Tcode );
fprintf(out,"Connected Edge : ₩n(Edge type-Block Number)₩n");
```

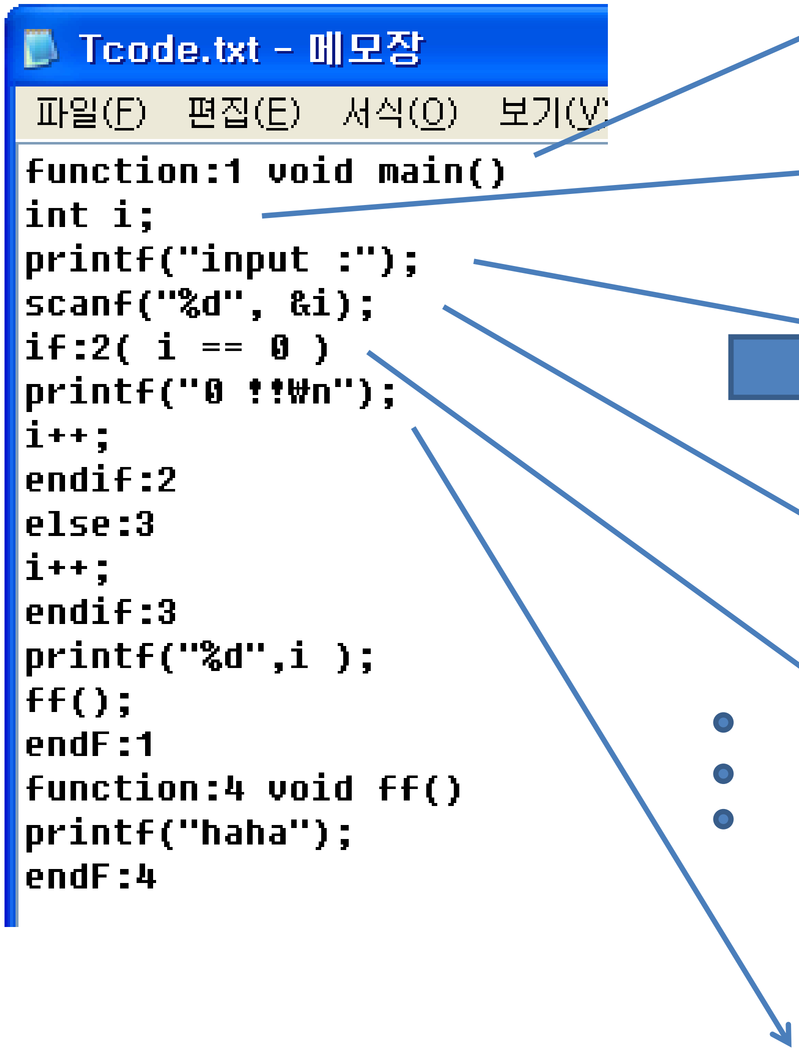
Structured Chart(Advanced)


a.txt - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
Tcode.txt - 메모장
파일(F) 편집(E) 서식(O) 보기(V)

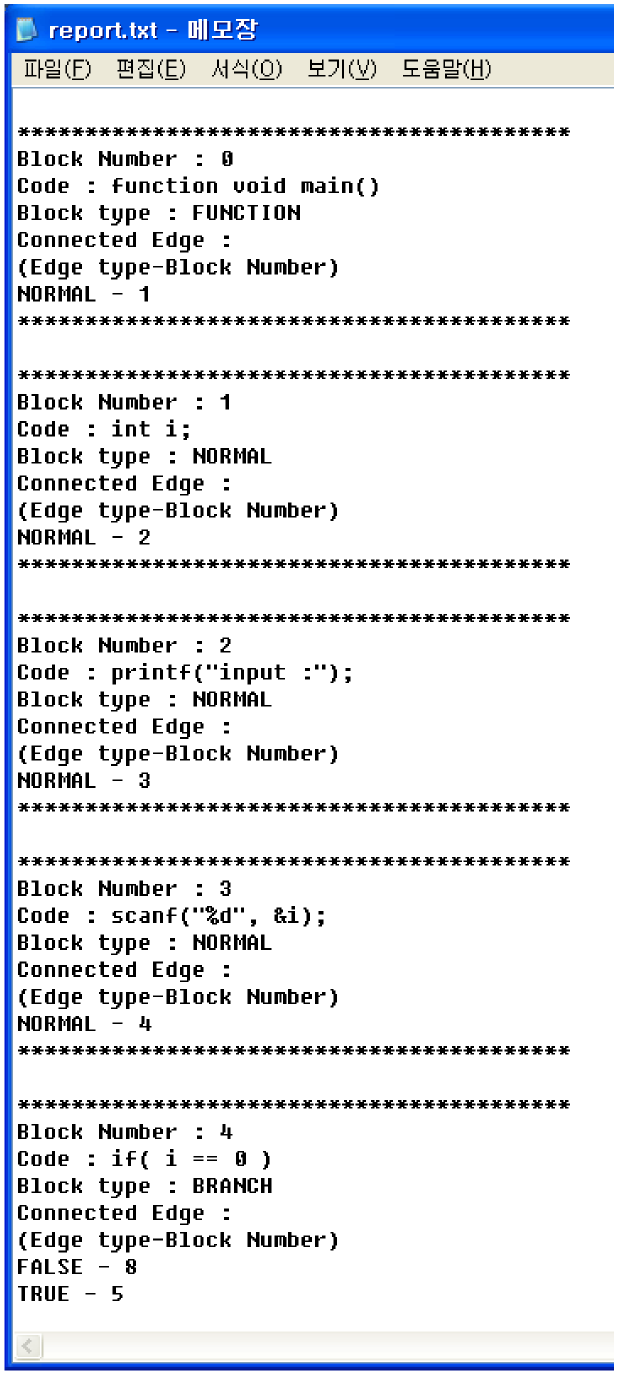
```
void main()
{
        int i;
        printf("input :");
        scanf("%d", &i);
        if( i == 0 )
        {
                printf("0 !!₩n");
                i++;
        }
        else
                i++;
        printf("%d",i );
        ff();
}

void ff()
{
        printf("haha");
}
```

```
function:1 void main()
int i;
printf("input :");
scanf("%d", &i);
if:2( i == 0 )
printf("0 !!₩n");
i++;
endif:2
else:3
i++;
endif:3
printf("%d",i );
ff();
endF:1
function:4 void ff()
printf("haha");
endF:4
```

Structured Chart(Advanced)

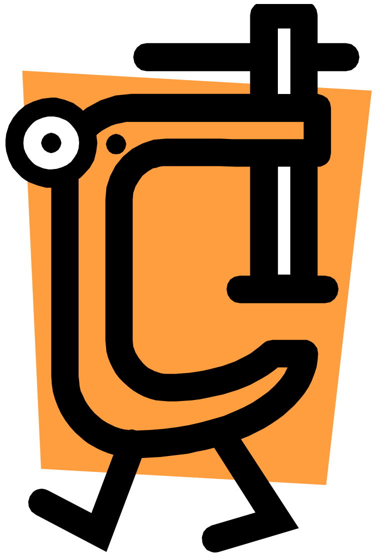


```
function:1 void main()
int i;
printf("input :");
scanf("%d", &i);
if:2( i == 0 )
printf("0 !!₩n");
i++;
endif:2
else:3
i++;
endif:3
printf("%d",i );
ff();
endF:1
function:4 void ff()
printf("haha");
endF:4
```



```
*****************************************
Block Number : 0
Code : function void main()
Block type : FUNCTION
Connected Edge :
(Edge type-Block Number)
NORMAL - 1
*****************************************

*****************************************
Block Number : 1
Code : int i;
Block type : NORMAL
Connected Edge :
(Edge type-Block Number)
NORMAL - 2
*****************************************

*****************************************
Block Number : 2
Code : printf("input :");
Block type : NORMAL
Connected Edge :
(Edge type-Block Number)
NORMAL - 3
*****************************************

*****************************************
Block Number : 3
Code : scanf("%d", &i);
Block type : NORMAL
Connected Edge :
(Edge type-Block Number)
NORMAL - 4
*****************************************

*****************************************
Block Number : 4
Code : if( i == 0 )
Block type : BRANCH
Connected Edge :
(Edge type-Block Number)
FALSE - 8
TRUE - 5
```

Example of execution



VPN
Internet
WAN
HQ
FW
FW
FW
FW
FW
VPN
DMZ
Server
lease line
SW
SW
SW
SW
Wi-Fi
Internet

Thank You!!