บทที่ 3

ออกแบบระบบ (System Design)

3.1 ภาพรวมและขอบเขตของระบบ

วิธีการโดยทั่วไปในการตรวจหาข้อผิดพลาดของโปรแกรมคอมพิวเตอร์ก็คือ การสร้างกรณีทดสอบขึ้นมาและเปรียบเทียบผลลัพธ์ที่ได้ในแต่ละกรณี

ในที่นี้เราจะพัฒนาเครื่องมือซึ่งจะเรียกว่า TEST_C ซึ่งเป็นเครื่องมือที่ใช้ทดสอบการทำงานของโปรแกรมคอมพิวเตอร์ในขณะที่ทำการทดสอบโปรแกรมด้วยข้อมูล ในการทดสอบนี้จะทำการตรวจสอบการทำงานในส่วนของคำสั่งที่เป็นคำสั่งการควบคุมสายงานอย่างมีเงื่อนไขของโปรแกรมภาษาซี และตรวจสอบการเรียกใช้งานแต่ละโมดุลในโปรแกรมภาษาซีในระหว่างการทดสอบโปรแกรมด้วยข้อมูล แล้วจัดเก็บผลลัพธ์ที่ได้ของการทดสอบแต่ละครั้ง เพื่อจัดทำเป็นสถิติและรายงานต่อไป

โครงสร้างของระบบ TEST_C เราแสดงได้ดังรูปที่ 3.1 ซึ่งจะประกอบด้วยส่วนของเครื่องมือ (tool), ส่วนของการติดต่อผู้ใช้งาน (user interface), ส่วนของบรรณาธิกร (editor), ส่วนของระบบปฏิบัติการ (dos shell), ส่วนของการแทรกตัวนับลงในรหัสต้นฉบับภาษาซี, ส่วนของการประมวลผล ซึ่งแต่ละส่วนจะอธิบายแยกย่อยกันไป

```
                :--> SETUP          :--> Give names of editor and path
                :
                :--> EDITOR         :--> Invoke editor
                :
                :--> DOS SHELL      :--> Active command.com
                :
                :                   :--> Input   -- C source code
 MAIN :-->      :--> INSTRUMENT     :--> Process -- add probes
                :                   :--> Output  -- instrumented program
                :                   :--> Compile Instrument code
                :
                :--> EXECUTE        :--> Execute Program
                :
                :--> ANALYSIS       :--> Generate reports to file
                :
                :--> CLEAR COUNTER  :--> Initialize Database
                :
                :--> HELP           :--> Display Help messages
```

รูปที่ 3.1 แสดงรูปแผนผังโครงสร้างของระบบ TEST_C

3.1.1 การติดต่อกับผู้ใช้งาน (user interface)

การติดต่อระหว่างผู้ใช้งานกับ TEST_C นั้น เราจะติดต่อโดยใช้แบบเชิงโต้ตอบ (interactive) บนหน้าจอ โดยจะมีเป็นรายการหัวข้อ (menu) ที่จะให้ผู้ใช้งานเลือก

3.1.2 การติดตั้งระบบ (setup)

จะให้ผู้ใช้งานกำหนดค่าต่างของระบบซึ่งได้แก่ ชื่อของตัวบรรณาธิกร, ชื่อของตัวแปลชุดคำสั่ง (compiler) และค่าตัวเลือกสำหรับตัวชุดคำสั่ง ซึ่งในที่นี้จะใช้ตัวแปลชุดคำสั่งเทอร์โบซี รุ่น 2.0

3.1.3 การทำบรรณาธิกร (editing facility)

จะให้ทำการแก้ไขโปรแกรม หรือ ข้อมูล ได้โดยไม่ต้องออกจากระบบของ test_c ซึ่งผู้ใช้สามารถเรียกเอาตัวบรรณาธิกร มาใช้งานได้ทันที ตามที่ได้กำหนดไว้ในหัวข้อการติดตั้งระบบ และตัวบรรณาธิกรที่กำหนดโดยปริยาย ถ้าไม่มีการเปลี่ยนแปลงคือ Q.EXE

3.1.4 การเรียกระบบปฏิบัติการชั่วคราว (DOS SHELL)

จะให้สามารถหยุดการทำงานของ TEST_C ชั่วคราวเพื่อเรียกใช้คำสั่งของระบบปฏิบัติการ และผู้ใช้สามารถกลับไปยังระบบ TEST_C เพื่อทำงานต่อในขณะที่หยุดระบบชั่วคราวได้

3.1.5 การแทรกตัวนับ (INSTRUMENTATION)

จะแทรกตัวนับโดยการวิธีการ กวาดตรวจ (scan) โปรแกรมรหัสต้นฉบับภาษาซี และแทรกตัวนับลงในตำแหน่งต้นของเส้นทางตัดสินใจ ซึ่งตัวนับจะทำหน้าที่รายงานถึงความถี่ของการแวะผ่าน (traversal) ในแต่ละเส้นทางตัดสินใจ และความถี่ในการเรียกใช้งานของแต่ละโมดุล

3.1.6 การประมวลผล (Execution)

ในการประมวลจะทำการทำหนดค่าแรกเริ่มของฐานข้อมูลที่ใช้สำหรับ TEST_C และทำการประมวลผลโปรแกรมภาษาซีที่ผ่านการแทรกตัวตรวจสอบการทำงานโปรแกรม และเก็บข้อมูลที่เป็นสถิติความถี่ต่างไว้สำหรับการทำรายงาน

3.1.7 การวิเคราะห์ (Analysis)

ในการวิเคราะห์ผลลัพธ์ที่ได้จะทำวิเคราะห์ข้อมูลเป็นรายงาน 3 รูปแบบคือ 1) รายงานความถี่ของการประมวลผลของเส้นทางตัดสินใจทั้งหมด ซึ่งผลลัพธ์ที่ได้จะเก็บไว้ที่แฟ้มข้อมูลชื่อ COMP_DDP.RPT 2) รายงานเส้นทางตัดสินใจที่ไม่เคยมีการประมวลผลเลยซึ่งผลลัพธ์จะเก็บไว้ที่แฟ้มข้อมูลชื่อ UN_TRAV.RPT 3) รายงานความถี่ของการเรียกใช้โมดุลซึ่งผลลัพธ์จะเก็บไว้ที่แฟ้มข้อมูลชื่อ MODULE_F.RPT

3.1.8 การเคลียร์ค่าตัวนับ (Clear counter)

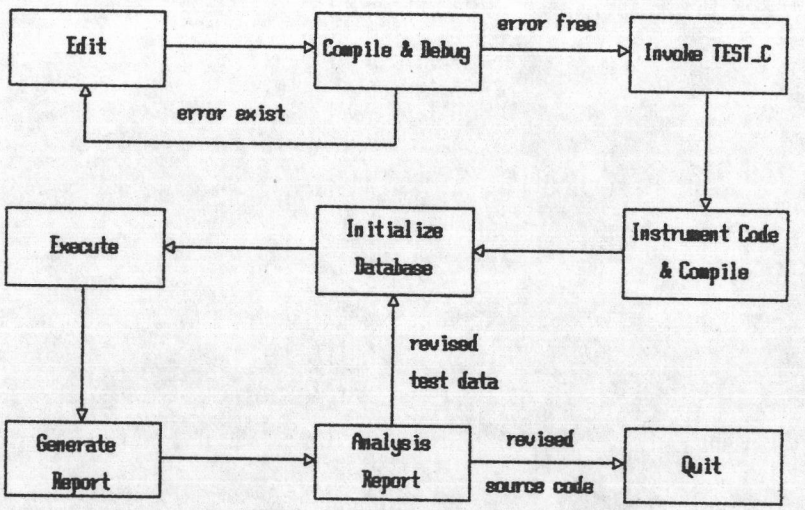
จะเป็นการกำหนดให้ค่าของตัวนับทุกตัวมีค่าเป็นศูนย์ ซึ่งเป็นค่าเริ่มต้น ใช้ในกรณีที่เราต้องการ จะเริ่มต้นประมวลผลของโปรแกรมที่ผ่านการแทรกตัวนับไว้แล้ว

3.1.9 การช่วยเหลือ (Help)

จะเป็นแบบเชิงโต้ตอบ คือจะมีข้อความสั้นๆ อธิบายในแต่ละหัวข้อพอสังเขปเพื่อให้เข้าใจและกำหนดค่าได้ถูกต้อง

3.2 ผังงานของระบบ (System Flow)

ขั้นตอนในการทำงานของ TEST_C แสดงได้ดังรูปที่ 3.2 ซึ่งเป็นรูปผังงานแสดงการทำงานของระบบซึ่งอธิบายขั้นตอนต่างๆ ได้ดังต่อไปนี้

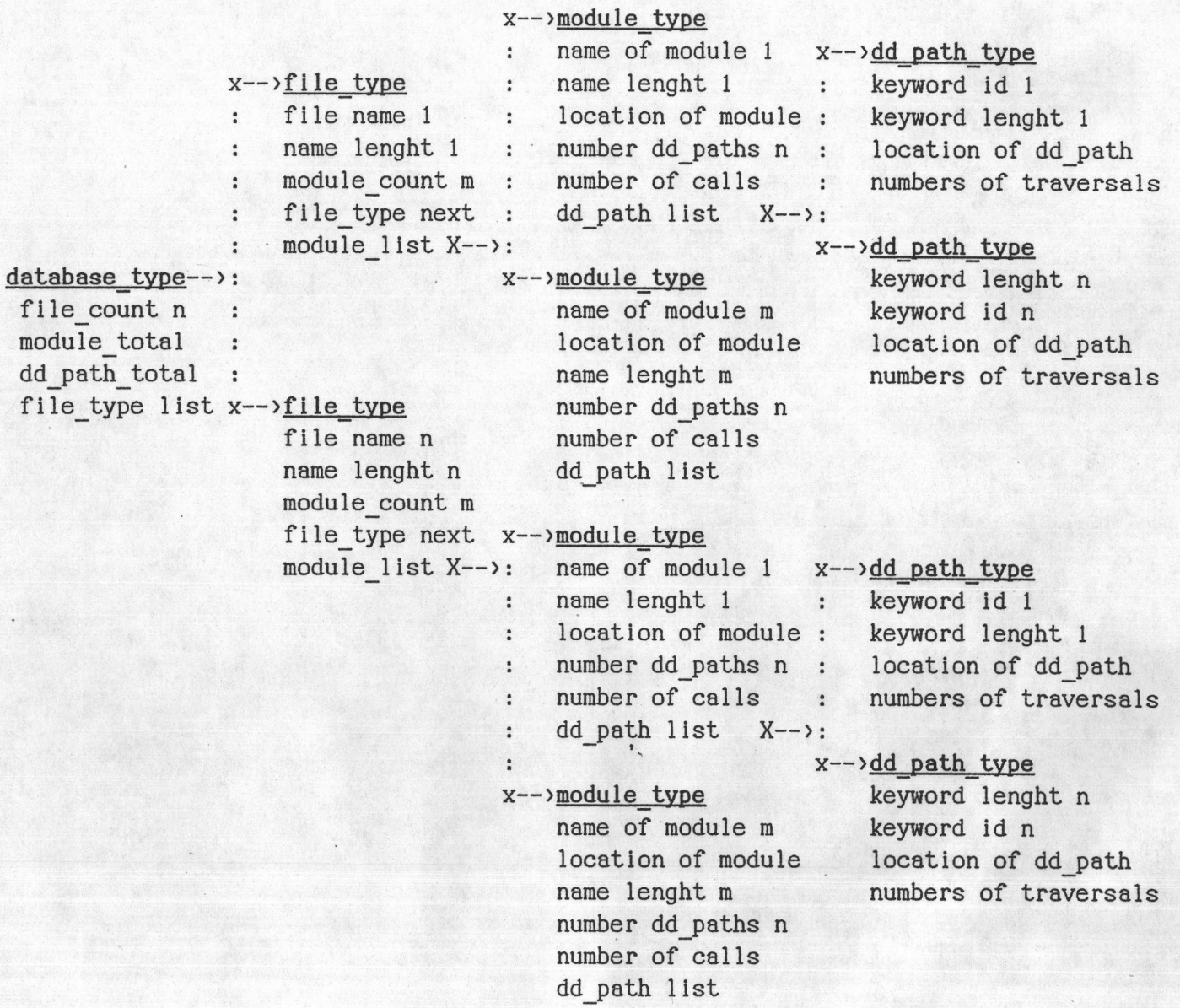


รูป 3.2 ผังงานแสดงขั้นตอนการทำงานระบบ TEST_C

1) โปรแกรมเมอร์เขียนโปรแกรมภาษาซี
2) ทำการแปลโปรแกรมและแก้ไขข้อผิดพลาด
3) ถ้าโปรแกรมยังมีข้อผิดพลาดอยู่ ให้ทำขั้นตอนที่ 1) และ 2) ใหม่
4) ผลที่ได้ไม่มีข้อผิดพลาดในโปรแกรม และพร้อมที่แทรกตัวนับ
5) เรียกใช้งานโปรแกรม TEST_C
6) แทรตัวนับลงในรหัสต้นฉบับภาษาซี และแปลชุดคำสั่ง
7) โปรแกรม TEST_C จะทำการกำหนดค่าเริ่มต้น
8) โปรแกรมรหัสต้นฉบับภาษาซีที่แทรกตัวนับไว้เรียบร้อย จะถูกประมวลผล
9) ข้อมูลผลลัพธ์รายงานต่างๆ จะถูกสร้างขึ้น
10) ผู้ใช้ทำการวิเคราะห์ดูรายงานผลลัพธ์ที่ได้
11) ถ้าต้องทำการทดสอบด้วยข้อมูลชุดใหม่ ให้ย้อนกลับไปเริ่มที่ข้อ 7
12) ถ้าต้องแก้ไขโปรแกรมรหัสต้นฉบับภาษาซีใหม่ให้ออกจาก TEST_C และเริ่มขั้นตอนที่ 1 ใหม่
13) จบการทำงาน

3.3 การออกแบบโครงสร้างข้อมูล และรายละเอียดข้อมูล

ฐานข้อมูลของ TEST_C ซึ่งจะถูกเรียกใช้โดย ตัวนับ ที่ถูกแทรกไว้ในรหัสต้นฉบับภาษาซี ในระหว่างที่ทดสอบโปรแกรมด้วยข้อมูลนั้น มีรูปแบบโครงสร้างของข้อมูลดังนี้ คือ database_type, module_type, และ dd_path_type ดังแสดงดังรูป 3.3 และรายละเอียดของแต่ละโครงสร้างจะอธิบายถัดไป



รูป 3.3 แสดงโครงสร้างของข้อมูล

3.3.1 รายละเอียดโครงสร้างของ database_type

เรากำหนดโครงสร้างของ database_type เป็นดังนี้

```
struct database_type {
        int file_count;
        int module_total;
        int dd_path_total;
        struct file_type *file_list;
}
```

ความหมายของตัวแปรในโครงสร้าง database_type

file_count

ชนิดข้อมูล : เลขจำนวนเต็ม (integer)

ค่าของข้อมูล : 0 - จำนวนสูงสุดของแฟ้มรหัสต้นฉบับภาษาซี

ค่าเริ่มต้น : 0

ความหมาย : เก็บค่าจำนวนของแฟ้มรหัสต้นฉบับภาษาซี

module_total

ชนิดข้อมูล : เลขจำนวนเต็ม

ค่าของข้อมูล : 0 - จำนวนสูงสุดของโมดุล

ค่าเริ่มต้น : 0

ความหมาย : เก็บค่าของจำนวนโมดุลทั้งหมด

dd_path_total

ชนิดข้อมูล : เลขจำนวนเต็ม

ค่าของข้อมูล : 0 - จำนวนสูงสุดของเส้นทางตัดสินใจ

ค่าเริ่มต้น : 0

ความหมาย : เก็บค่าจำนวนของเส้นทางเดินตัดสินใจทั้งหมด

file_list

ชนิดข้อมูล : ดัชนีชี้ไปยังโครงสร้างข้อมูลชนิด file_type

ค่าของข้อมูล : มาตรฐานตามค่าชนิดดัชนี

ค่าเริ่มต้น : ว่าง (NULL)

ความหมาย : เป็นดัชนีซึ่งชี้ไปยังโครงสร้างข้อมูลตัวแรกของ file_type

3.3.2 รายละเอียดโครงสร้างของ file_type

เรากำหนดโครงสร้างของ file_type เป็นดังนี้

```
struct file_type {
        char *name;
        int name_len;
        int num_of_module;
        struct module_type *module_list;
        struct file_type *next;
}
```

ความหมายของตัวแปรในโครงสร้าง file_type

name

ชนิดข้อมูล : ดัชนีของข้อมูลตัวอักษร (character pointer)
ค่าของข้อมูล : มาตรฐานตามค่าชนิดดัชนี
ค่าเริ่มต้น : ว่าง (NULL)
ความหมาย : เป็นดัชนีชี้ไปยังชื่อของแฟ้มรหัสต้นฉบับ

name_len

ชนิดข้อมูล : เลขจำนวนเต็ม
ค่าของข้อมูล : 0 - จำนวนความยาวของตัวอักษร
ค่าเริ่มต้น : 0
ความหมาย : เก็บค่าความยาวของชื่อแฟ้มรหัสต้นฉบับ

module_count

ชนิดข้อมูล : เลขจำนวนเต็ม (integer)
ค่าของข้อมูล : 0 - จำนวนสูงสุดของโมดุลในแฟ้มข้อมูล
ค่าเริ่มต้น : 0
ความหมาย : เก็บค่าจำนวนโมดุลทั้งหมดที่มีอยู่ในแฟ้มรหัสต้นฉบับภาษาซี

module_list

ชนิดข้อมูล : ดัชนีชี้ไปยังโครงสร้างข้อมูล module_type
ค่าของข้อมูล : มาตรฐานตามค่าชนิดดัชนี
ค่าเริ่มต้น : ว่าง (NULL)
ความหมาย : เป็นดัชนีซึ่งชี้ไปยังโครงสร้างที่บอกถึงสารสนเทศ ของคำสั่งควบคุมสายงานในแต่ละโมดุล

next

ชนิดข้อมูล : ดัชนีชี้ไปยังโครงสร้างข้อมูล file_type

ค่าของข้อมูล : มาตรฐานตามค่าชนิดดัชนี

ค่าเริ่มต้น : ว่าง (NULL)

ความหมาย : เป็นดัชนีซึ่งชี้ไปยังโครงสร้างข้อมูล file_type ตัวถัดไป มีค่าเป็น NULL ถ้าสิ้นสุดข้อมูล

3.3.3 โครงสร้างของ module_type

เรากำหนดโครงสร้างของ module_type เป็นดังนี้

```
struct module_type {
        char *name;
        int name_len;
        int line_number;
        int num_dd_path;
        long call_frequency;
        struct dd_path_type *dd_path_list;
        struct module_type *next;
}
```

ความหมายของตัวแปรในโครงสร้าง module_type

name

ชนิดข้อมูล : ดัชนีของข้อมูลตัวอักษร (character pointer)

ค่าของข้อมูล : มาตรฐานตามค่าชนิดดัชนี

ค่าเริ่มต้น : ว่าง (NULL)

ความหมาย : เป็นดัชนีชี้ไปยังชื่อของโมดุล

name_len

ชนิดข้อมูล : เลขจำนวนเต็ม

ค่าของข้อมูล : 0 - จำนวนความยาวของตัวอักษร

ค่าเริ่มต้น : 0

ความหมาย : เก็บค่าความยาวของชื่อโมดุล

line_number

ชนิดข้อมูล : เลขจำนวนเต็ม (integer)
ค่าของข้อมูล : 0 - จำนวนสูงสุดของบรรทัดที่อยู่ในแฟ้มข้อมูล
ค่าเริ่มต้น : 0
ความหมาย : หมายเลขบรรทัดที่อยู่ในแฟ้มข้อมูลซึ่งเป็นบรรทัดแรกของโมดุล

num_dd_path

ชนิดข้อมูล : เลขจำนวนเต็ม (integer)
ค่าของข้อมูล : 0 - จำนวนสูงสุดของคำสั่งควบคุมสายงานที่อยู่ในแฟ้มข้อมูล
ค่าเริ่มต้น : 0
ความหมาย : จำนวนคำสั่งควบคุมสายงานทั้งหมดในโมดุล

call_frequency

ชนิดข้อมูล : เลขจำนวนเต็ม (long)
ค่าของข้อมูล : 0 - จำนวนสูงสุดของการเรียกโมดุลในขณะทดสอบโปรแกรม
ค่าเริ่มต้น : 0
ความหมาย : จำนวนครั้งในการเรียกโมดุลในขณะที่ประมวลผลโปรแกรม

dd_path_list

ชนิดข้อมูล : ดัชนีชี้ไปยังโครงสร้างข้อมูล dd_path_type
ค่าของข้อมูล : มาตรฐานตามค่าชนิดดัชนี
ค่าเริ่มต้น : ว่าง (NULL)
ความหมาย : เป็นดัชนีซึ่งชี้ไปยังโครงสร้างที่บอกถึงสารสนเทศ ของคำสั่งควบคุมสายงานในแต่ละโมดุล

next

ชนิดข้อมูล : ดัชนีชี้ไปยังโครงสร้างข้อมูล module_type
ค่าของข้อมูล : มาตรฐานตามค่าชนิดดัชนี
ค่าเริ่มต้น : ว่าง (NULL)
ความหมาย : เป็นดัชนีซึ่งชี้ไปยังโครงสร้างข้อมูล module_type ตัวถัดไป มีค่าเป็น NULL ถ้าสิ้นสุดข้อมูล

3.3.4 โครงสร้างของ dd_path_type

เรากำหนดโครงสร้างของ dd_path_type เป็นดังนี้

```
struct dd_path_type {
        char *keyword;
        int keyword_len;
        int line_number;
        long traversal_freq;
        struct dd_path_type *next;
}
```

ความหมายของตัวแปรใน dd_path_type

keyword

ชนิดข้อมูล : ดัชนีของข้อมูลตัวอักษร (character pointer)
ค่าของข้อมูล : มาตรฐานตามค่าชนิดดัชนี
ค่าเริ่มต้น : ว่าง (NULL)
ความหมาย : เป็นดัชนีชี้ไปยังชื่อของคำสั่งควบคุมสายงานเช่น if, while

name_len

ชนิดข้อมูล : เลขจำนวนเต็ม
ค่าของข้อมูล : 0 - จำนวนความยาวของตัวอักษร
ค่าเริ่มต้น : 0
ความหมาย : เก็บค่าความยาวของ keyword

line_number

ชนิดข้อมูล : เลขจำนวนเต็ม (integer)
ค่าของข้อมูล : 0 - จำนวนสูงสุดของบรรทัดที่อยู่ในแฟ้มข้อมูล
ค่าเริ่มต้น : 0
ความหมาย : เลขที่บรรทัดในแฟ้มข้อมูลภาษาซี ของคำสั่งควบคุมสายงาน

traversal_freq

ชนิดข้อมูล : เลขจำนวนเต็ม (long)
ค่าของข้อมูล : 0 - จำนวนสูงสุดของค่าจำนวนเต็ม
ค่าเริ่มต้น : 0
ความหมาย : จำนวนครั้งในการประมวลผลในระหว่างการทดสอบโปรแกรม

next

ชนิดข้อมูล : ดัชนีชี้ไปยังโครงสร้างข้อมูล dd_path_type

ค่าของข้อมูล : มาตรฐานตามค่าชนิดดัชนี

ค่าเริ่มต้น : ว่าง (NULL)

ความหมาย : เป็นดัชนีซึ่งชี้ไปยังโครงสร้างข้อมูล dd_path_type ตัวถัดไป มีค่าเป็น NULL ถ้าสิ้นสุดข้อมูล

3.4 ออกแบบโครงสร้างโมดุล และรายละเอียดโมดุล

ลักษณะการทำงานในแต่ละส่วนของโมดุลนั้นเราจะมี input, process, output ดังที่จะกล่าวต่อไปนี้ ในบทนี้จะออกแบบรายละเอียดแต่ละโมดุลใน TEST_C แต่ละโมดุลที่อธิบายจะอยู่ในรูปแบบดังต่อไปนี้

ชื่อโมดุล : ชื่อของโมดุล

หมายเลขโมดุล : หมายเลขที่ให้ไว้สำหรับโมดุล

รับเข้า (input) : ข้อมูลเข้าสำหรับโมดุล

ส่งออก (output): ผลลัพธ์ที่ได้จากโมดุล

คำอธิบาย : เป็นคำอธิบายทั่วไปเกี่ยวกับโมดุล

เรียก : เรียกใช้โมดุลอื่นๆ อะไรบ้าง

ถูกเรียกโดย : ถูกเรียกใช้งานจากโมดุลใดบ้าง

การทดสอบ : วิธีการที่จะทดสอบโมดุลให้ถูกต้อง

ข้อจำกัด : ข้อจำกัดเงื่อนไขต่างๆ เกี่ยวกับโมดุล

รหัสเทียม : เป็นลักษณะของโปรแกรมระดับสูงที่บอกการทำงานของโมดุล

3.4.1 main()

ชื่อโมดุล:	main
หมายเลขโมดุล:	1
รับเข้า:	ไม่มี
ส่งออก:	ไม่มี
คำอธิบาย:	แสดงบทนำและเมนูรายการบนหน้าจอ แล้วผู้ใช้งานเลือกหัวข้อที่ต้องการ

เรียก:	display_logo()
	display_main_menu()
	system_setup()
	shell_to_editor()
	shell_to_dos()
	instrument_source_code()
	compile_instrument_code()
	execution()
	write_DB_file()
	read_DB_file();
	analysis_menu()
	clear_head_data();
	init_DB_file()
	system_help()

ถูกเรียกโดย:	ไม่มี
การทดสอบ:	-
ข้อจำกัด:	ไม่มี

pseudocode:

```
int main()
{
  display_logo();
  display_main_menu();
  while (get choice) {
     switch (choice) {
        case 1: system_setup();
                break;
        case 2: execute_editor();
                break;
        case 3: shell_to_dos();
                break;
        case 4: instrument_code();
                break;
        case 5: execute_instrument_code();
                break;
        case 6: analysis_menu();
                break;
        case 7: init_DB_file();
                break;
        case 8: get_help();
                break;
        case 9: exit(); /* exit program */
                break;
        default: prompt user for next input and
                take no other action;
     }
  } while (TRUE);
}
```

3.4.2 display_logo()

ชื่อโมดุล: display_logo
หมายเลขโมดุล: 2
รับเข้า: ไม่มี
ส่งออก: ไม่มี
คำอธิบาย: แสดงชื่อโปรแกรม ชื่อผู้พัฒนาโปรแกรมบนหน้าจอ
เรียก: ไม่มี
ถูกเรียกโดย: main
การทดสอบ: ดูผลลัพธ์ทางหน้าจอ
ข้อจำกัด: ไม่มี

pseudocode:

```
void display_logo()
{
    display title "TEST_C" in the top program;
    display developer names;
    display "Press any key to continue";
    wait for input from user;
}
```

3.4.3 display_main_menu()

ชื่อโมดุล: display_main_menu
หมายเลขโมดุล: 3
รับเข้า: ไม่มี
ส่งออก: ไม่มี
คำอธิบาย: แสดงรายการหัวข้อต่างๆ บนหน้าจอ
เรียก: ไม่มี
ถูกเรียกโดย: main
การทดสอบ: ดูผลลัพธ์ทางหน้าจอ
ข้อจำกัด: ไม่มี

pseudocode:

```
void display_main_menu()
{
     display title "TEST_C" in the top screen;
     display the menu choices;
}
```

3.4.4 system_setup()

ชื่อโมดุล: system_setup
หมายเลขโมดุล: 4
รับเข้า: ไม่มี
ส่งออก: ไม่มี
คำอธิบาย: หัวข้อที่ต้องการ
เรียก: ไม่มี
ถูกเรียกโดย: main
การทดสอบ: -
ข้อจำกัด: ไม่มี

pseudocode:

```
void system_setup()
{
     display system setup menu;
     while (select chioce) {
          switch (chioce) {
               case 'e':
               case 'E':
                    display title setup editor;
                    get new editor name;
                    break;
               case 'p':
               case 'P':
                    display title setup path;
                    get new path;
                    break;
          }
     }
}
```

3.4.5 shell_to_editor()

ชื่อโมดุล: shell_to_editor
หมายเลขโมดุล: 5
รับเข้า: ไม่มี
ส่งออก: ไม่มี
คำอธิบาย: เรียกประมวลผลบรรณาธิกรชั่วคราว
เรียก: ไม่มี
ถูกเรียกโดย: main
การทดสอบ: -
ข้อจำกัด: ไม่มี

pseudocode:

```
void shell_to_editor()
{
     display title in the top screen;
     display "Exit editor noarmally to return to Main Menu";
     system(test_c_editor);
}
```

3.4.6 shell_to_dos()

ชื่อโมดุล: shell_to_dos()
หมายเลขโมดุล: 6
รับเข้า: ไม่มี
ส่งออก: ไม่มี
คำอธิบาย: ประมวลผลคำสั่งระบบปฏิบัติการชั่วคราว
เรียก: ไม่มี
ถูกเรียกโดย: main
การทดสอบ: -
ข้อจำกัด: ไม่มี

pseudocode:

```
void shell_to_dos()
{
     display title "DOS-Shell" in the top screen;
     display "Type  'EXIT' to return to Main Menu";
     system("command.com");
}
```

3.4.7 instrument_source_code()

ชื่อโมดุล: instrument_source_code
หมายเลขโมดุล: 7
รับเข้า: แฟ้มรหัสต้นฉบับภาษาซีที่จะแทรกฟังก์ชั่นตัวนับ
ส่งออก: แฟ้มรหัสต้นฉบับภาษาซีที่แทรกฟังก์ชั่นตัวนับแล้ว และมีส่วนขยายชื่อใหม่เป็น *.TCC
คำอธิบาย: แทรกฟังก์ชั่นตัวนับในแฟ้มรหัสต้นฉบับภาษาซี
เรียก: insert_probe
ถูกเรียกโดย: main
การทดสอบ: ตรวจดูแฟ้มรหัสภาษาซี *.TCC ที่ได้ว่าถูกต้องหรือไม่
ข้อจำกัด: ไม่มี

pseudocode:

```
void instrument_source_code()
{
     n = 0;
     display title instrument source code;
     while ( end of file name to instrument ) {
          get file name n to instrument;
          n++;
     }
     open file TESTC.PRJ;
     write TESTC.TCC to TESTC.PRJ;
     for ( n=0; n < num of file; n++) {
          insert probe to file name n,
          and write output to new file *.TCC;
          write new file *.TCC TO TESTC.PRJ;
     }
     close file TEST.PRJ;
     system("tc test.prj");
}
```

3.4.8 write_DB_file()

ชื่อโมดุล: write_DB_file
หมายเลขโมดุล: 8
รับเข้า: ไม่มี
ส่งออก: แฟ้มข้อมูลชื่อ TEST_C.DBF
คำอธิบาย: แสดงบทนำและเมนูรายการบนหน้าจอ แล้วผู้ใช้งานเลือกหัวข้อที่ต้องการ
เรียก: ไม่มี
ถูกเรียกโดย: main
การทดสอบ: ดูว่ามีแฟ้มข้อมูลชื่อ TEST_C.DBF ถูกสร้างขึ้นหรือไม่
ข้อจำกัด: จะต้องมีเนื้อที่ในแผ่นจานแม่เหล็กพอที่จะเก็บข้อมูล

pseudocode:

```
void write_DB_file()
{
     open output file "TEST_C.DBF";
     write head_data;
     for ( i=0; i<num_of_file; i++ ) {
          write struct file_type;
          for ( j=0; j<num_of_module; j++ ) {
               write struct module_type;
               for ( k=0; k<num_of_dd_path; k++ ) {
                    write struct dd_path_type;
                    get next dd_path_type;
               }
               get next module_type;
          }
          get next file_type;
     }
}
```

3.4.9 compile_instrument_code()

ชื่อโมดุล:	compile_instrument_code
หมายเลขโมดุล:	9
รับเข้า:	แฟ้มข้อมูลโปรเจคไฟล์ชื่อ TESTC.PRJ
ส่งออก:	จะได้โปรแกรมที่แทรกตัวนับไว้ชื่อ TESTC.EXE
คำอธิบาย:	เรียกประมวลผล เทอร์โบซี รุ่น 2.0 เพื่อแปลชุดคำสั่งโดยใช้โปรเจคไฟล์ชื่อ TESTC.PRJ
เรียก:	ไม่มี
ถูกเรียกโดย:	main
การทดสอบ:	-
ข้อจำกัด:	หน่วยความจำของระบบ ถ้าไม่พอสำหรับการแปลชุดคำสั่งให้จบการทำงาน ออกไปยังระบบปฏิบัติการก่อน แล้วแปลชุดคำสั่งใหม่โดยใช้โปรเจคไฟล์ TESTC.PRJ แล้วค่อยกลับมาทำต่อไป

pseudocode:

```
void compile_instrument_code()
{
     display title compile instrument code;
     display "Exit 'TC' normally to return to main menu";
     system("tc testc.prj");
}
```

3.4.10 execution()

ชื่อโมดุล: execution
หมายเลขโมดุล: 10
รับเข้า: เป็นชื่อโปรแกรม และอากิวเมนท์ เช่น testc ftype.c
ส่งออก: แล้วแต่ลักษณะการทำงานของโปรแกรมที่นำมาแทรกตัวนับ และจะได้ข้อมูลเกี่ยวกับตัวนับในแฟ้ม TEST_C.DBF
คำอธิบาย: ประมวลผลโปรแกรมที่ผ่านการแทรกตัวนับไว้แล้ว
เรียก: ไม่มี
ถูกเรียกโดย: main
การทดสอบ: ตรวจสอบดูว่าสามารถประมวลผลโปรแกรมที่ต้องการได้หรือไม่
ข้อจำกัด: ไม่มี

pseudocode:

```
void execution()
{
     display title "EXECUTE" in the top screen;
     display "Enter command line arguments for test case 1";
     get command argument to test_c_argument;
     system("test_c_arguments");
}
```

3.4.11 read_DB_file();

ชื่อโมดุล: read_DB_file
หมายเลขโมดุล: 11
รับเข้า: แฟ้มข้อมูลชื่อ TEST_C.DBF
ส่งออก: ค่า 1 มีแฟ้มข้อมูลและข้อมูลถูกต้อง, นอกนั้นจะส่งออกค่า 0
คำอธิบาย: อ่านแฟ้มข้อมูล TEST_C.BDF ซึ่งเป็นข้อมูลของตัวนับต่างๆของโปรแกรมที่ได้ผ่านการทำ instrument_source_code
เรียก: ไม่มี
ถูกเรียกโดย: main
การทดสอบ: สร้างรายงานที่ได้จากการอ่านข้อมูล
ข้อจำกัด: ไม่มี

pseudocode:

```
int read_DB_file()
{
    open input file "TEST_C.DBF";
    read head_data from TEST_C.DBF;
    for ( i=0; i<num_of_file; i++ ) {
        read struct file_type from TEST_C.DBF;
        for ( j=0; j<num_of_module; j++ ) {
            read struct module_type from TEST_C.DBF;
            for ( k=0; k<num_of_dd_path; k++ ) {
                read struct dd_path_type from TEST_C.DBF;
            }
        }
    }
}
```

3.4.12 analysis_menu()

ชื่อโมดุล: analysis_menu
หมายเลขโมดุล: 12
รับเข้า: แฟ้มข้อมูล TEST_C.DBF
ส่งออก: แฟ้มข้อมูล COMP_DDP.RPT, UN_TRAV.RPT, MODULE_F.RPT
คำอธิบาย: นำข้อมูลที่ได้มาจัดทำเป็นรายงาน 3 รายงาน
เรียก: complete_ddp_report(), untraversal_ddp_report(), module_freq_report()
ถูกเรียกโดย: main
การทดสอบ: ดูผลลัพธ์จากรายงานทั้ง 3 ว่าถูกต้องหรือไม่
ข้อจำกัด: ไม่มี

pseudocode:

```
void analysis_manu()
{
     display analysis menu;
     while ( select choice ) {
          switch ( choice ) {
               case 'C': complete_ddp_report();
                    break;
               case 'U': untraversal_ddp_report();
                    break;
               case 'M': module_freq_report();
                    break;
               case 'X':
                    return();
          }
     }
}
```

3.4.13 clear_head_data();

ชื่อโมดุล: clear_head_data
หมายเลขโมดุล: 13
รับเข้า: ไม่มี
ส่งออก: ไม่มี
คำอธิบาย: เคลียร์ค่าของข้อมูลใน head_data
เรียก: ไม่มี
ถูกเรียกโดย: main
การทดสอบ: -
ข้อจำกัด: ไม่มี

pseudocode:

```
void clear_head_data()
{
     for ( i=0; i<num_of_file; i++ ) {
          for ( j=0; j<num_of_module; j++ ) {
               for ( k=0; k<num_of_dd_path; k++ ) {
                    save pointer to next dd_path_type;
                    free old struct dd_path_type;
                    get next dd_path_type;
               }
               save pointer to next module_type;
               free old struct module_type;
               get next module_type;
          }
          save pointer to next file_type;
          free old struct file_type;
          get next file_type;
     }
}
```

3.4.14 init_DB_file()

ชื่อโมดุล: init_DB_file
หมายเลขโมดุล: 14
รับเข้า: แฟ้มข้อม TEST_C.DBF
ส่งออก: แฟ้มข้อม TEST_C.DBF
คำอธิบาย: กำหนดค่าของตัวนับทุกตัวในแฟ้ม TEST_C.DBF ให้เป็นศูนย์
เรียก: read_DB_file();
write_db_file();
clear_head_data();
ถูกเรียกโดย: main
การทดสอบ: สร้างรายงานทั้งสามรายงานเพื่อดูว่าค่าตัวนับเป็นศูนย์หรือไม่
ข้อจำกัด: ต้องมีแฟ้ม TEST_C.DBF ซึ่งเป็นข้อมูลของตัวนับ

pseudocode:

```
void init_database()
{
    display title "Initial data base file";
    clear_head_data();
    read_DB_file();
    for ( i=0; i<num_of_file; i++ ) {
        for ( j=0; j<num_of_module; j++ ) {
            for ( k=0; k<num_of_dd_path; k++ ) {
                set dd_path counter to zero;
                get next dd_path_type;
            }
            set module counter to zero;
            get next module_type;
        }
        get next file_type;
    }
    write_DB_file();
}
```

3.4.15 system_help()

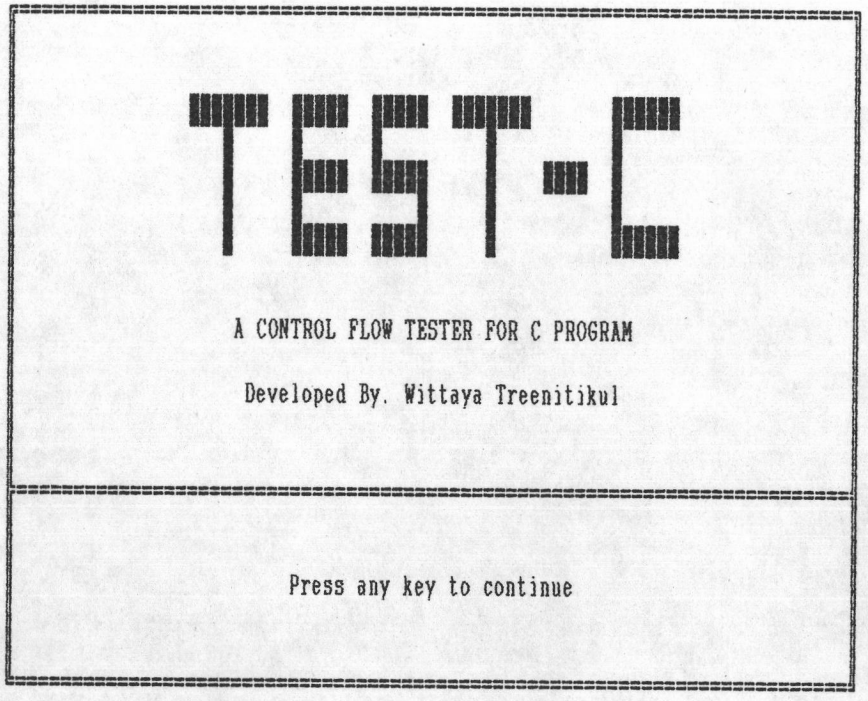
ชื่อโมดุล: system_help
หมายเลขโมดุล: 15
รับเข้า: ไม่มี
ส่งออก: ไม่มี
คำอธิบาย: อธิบายหัวข้อแต่ละหัวข้อใน main menu
เรียก: display_system_help();
display_help_system_setup();
display_help_editor();
display_help_dos_shell();
display_help_instrumentation();
display_help_execution();
display_help_analysis();
ถูกเรียกโดย: main
การทดสอบ: ดูผลลัพธ์ที่แสดงบนหน้าจอ
ข้อจำกัด: ไม่มี

pseudocode:

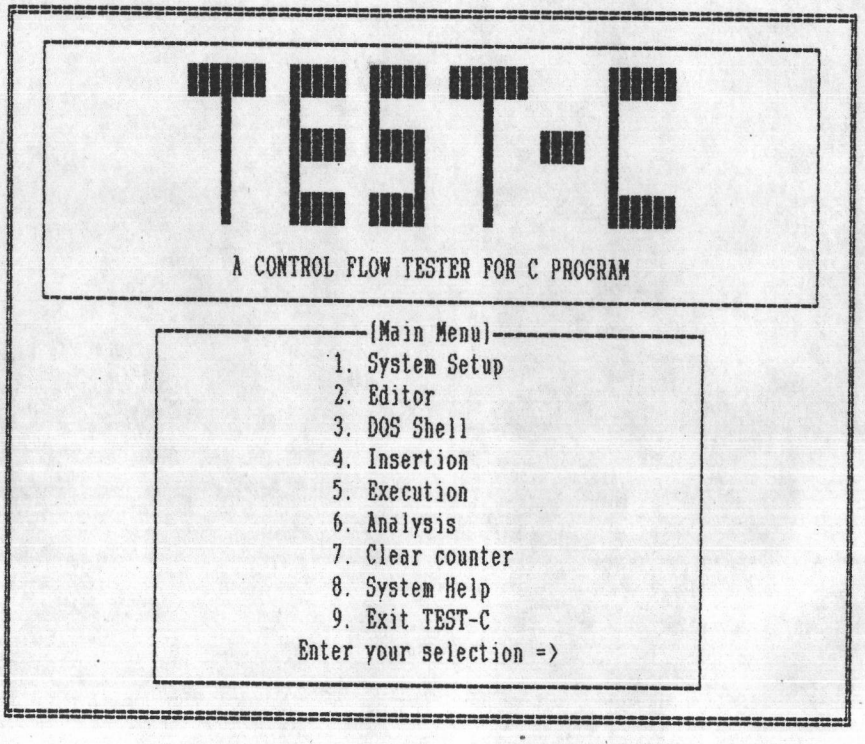
```
void system_help()
{
     display help menu;
     while( select choice ) {
          display help message from user select choice;
          wait for user respone;
          display help menu;
     }
}
```

3.5 **ออกแบบจอภาพ**

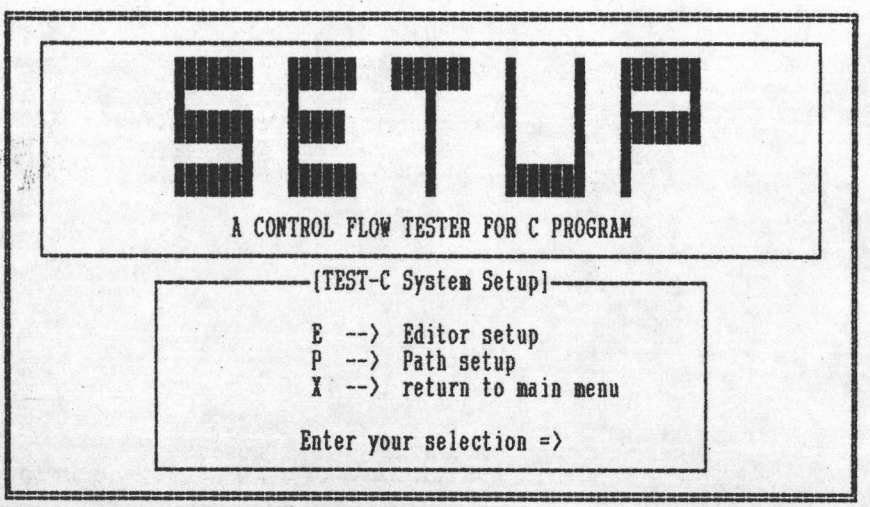
หน้าตาจอภาพที่ใช้แสดงในเครื่องมือ TEST_C ซึ่งออกแบบไว้นั้นแสดงได้ดังรูปที่ 3.5.1 ถึงรูปที่ 3.5.22



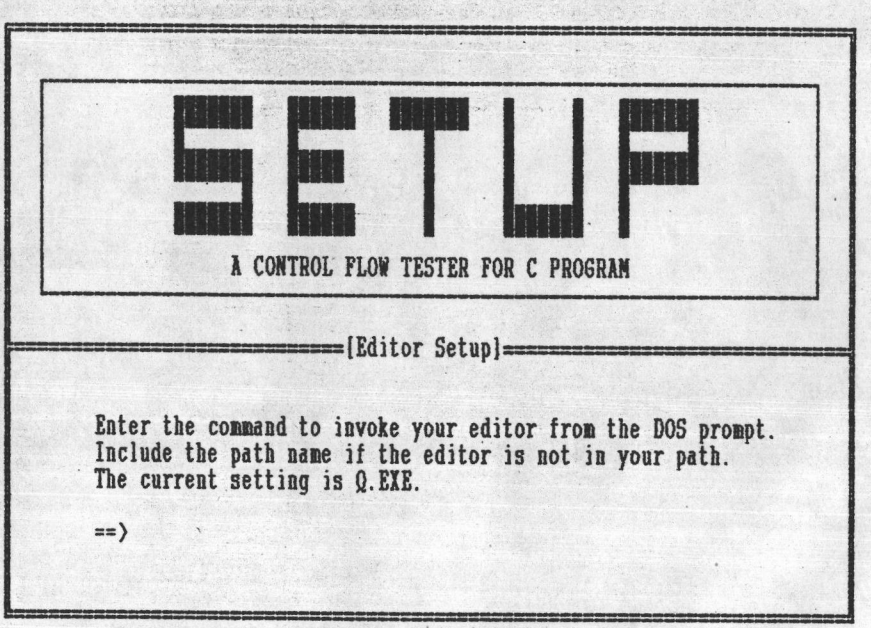
รูป 3.5.1 แสดงรูปจอภาพของ Introduction



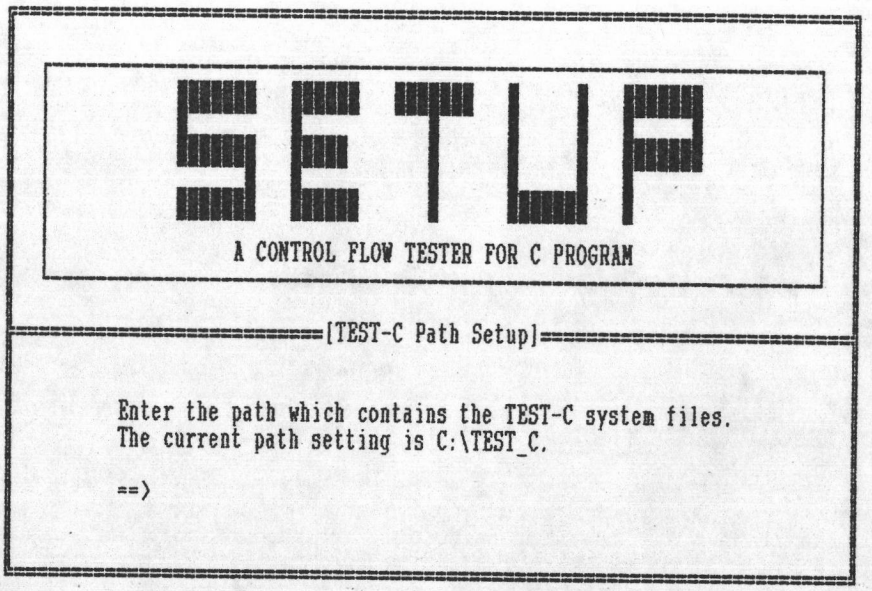
รูป 3.5.2 แสดงรูปจอภาพของ Main Menu



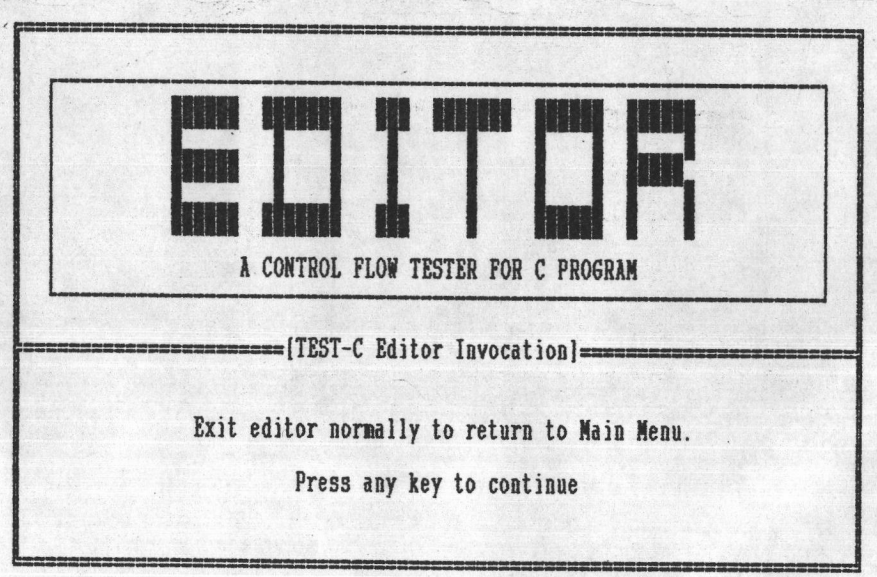
รูป 3.5.3 แสดงรูปจอภาพของ System Setup Menu



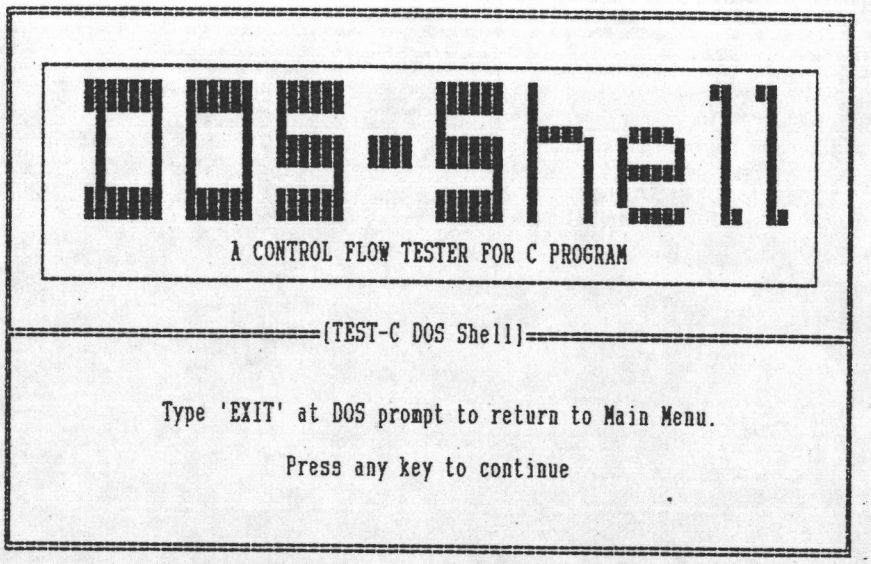
รูป 3.5.4 แสดงรูปจอภาพของ Editor Setup



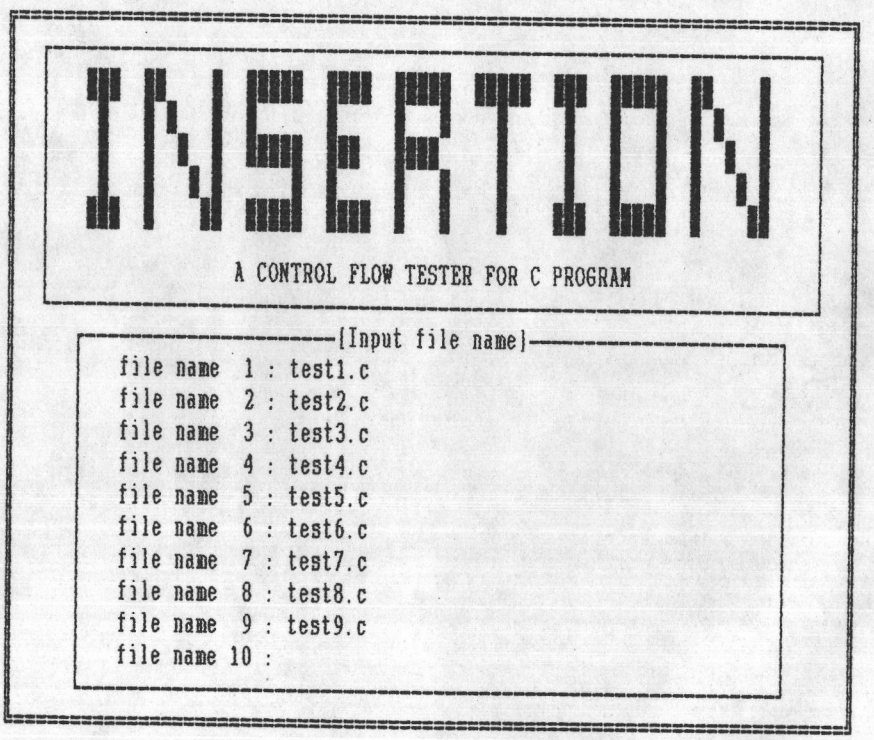
รูป 3.5.5 แสดงรูปจอภาพของ Path Setup



รูป 3.5.6 แสดงรูปจอภาพของการเรียกใช้ Editor

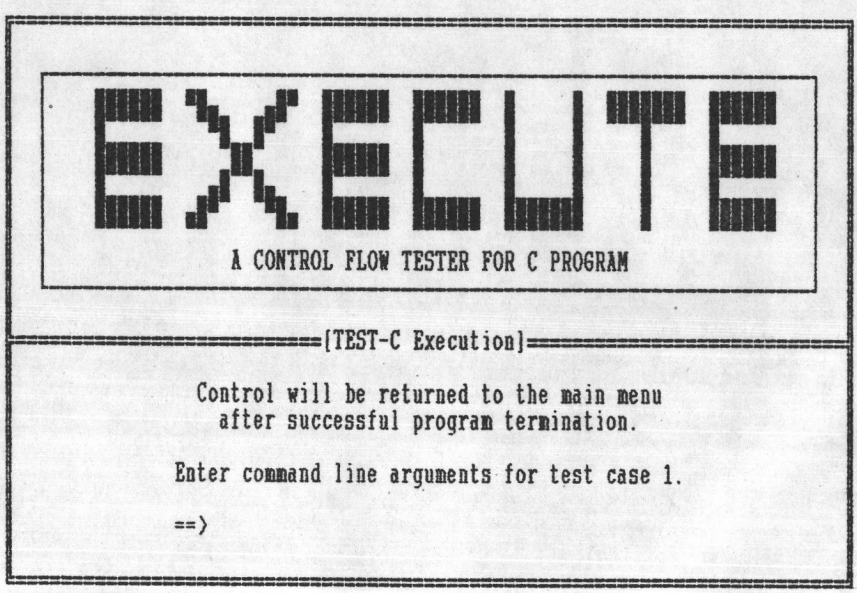


รูป 3.5.7 แสดงรูปจอภาพของการเรียกใช้ Dos Shell

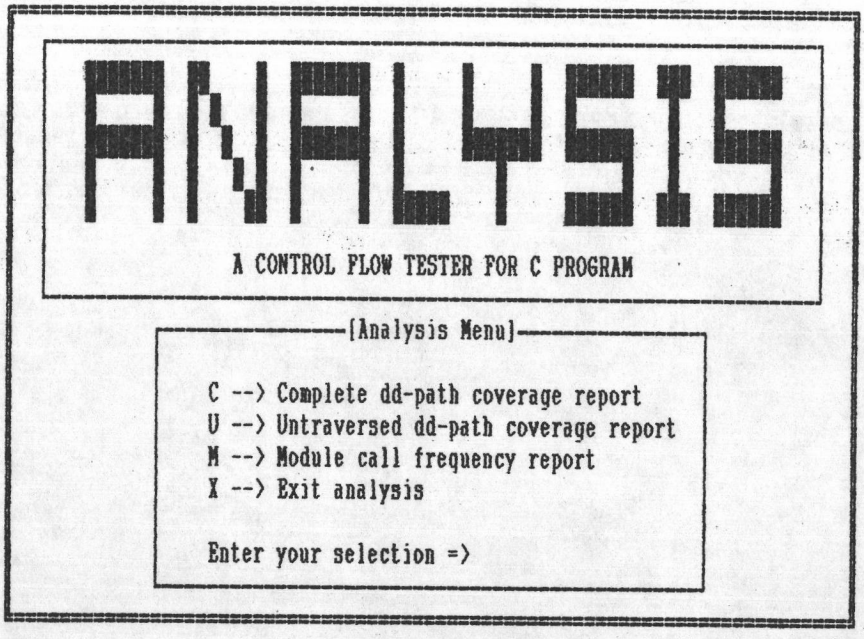


รูป 3.5.8 แสดงรูปจอภาพของ Insertion

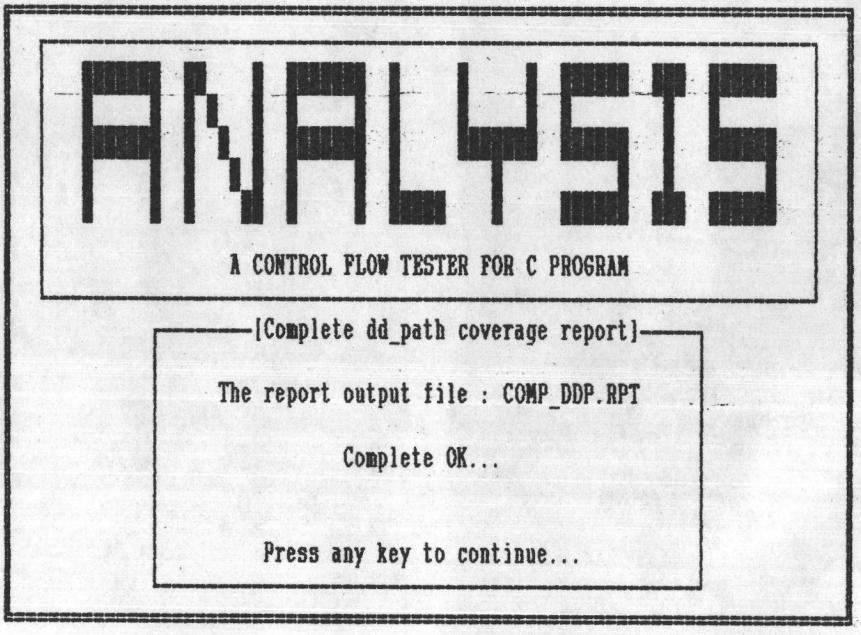
รูป 3.5.9 แสดงรูปจอภาพของการ Compile



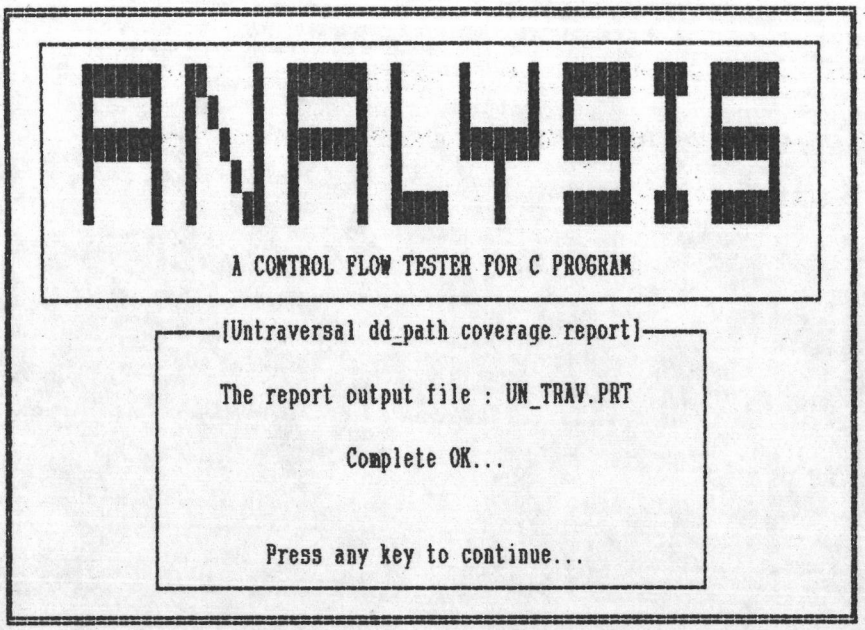
รูป 3.5.10 แสดงรูปจอภาพของ Execution



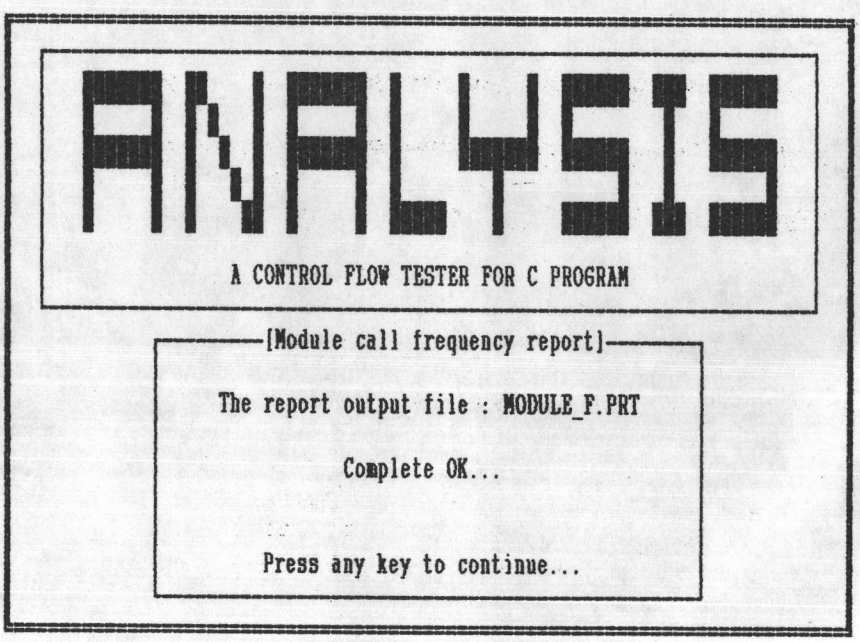
รูป 3.5.11 แสดงรูปจอภาพของ Analysis Menu



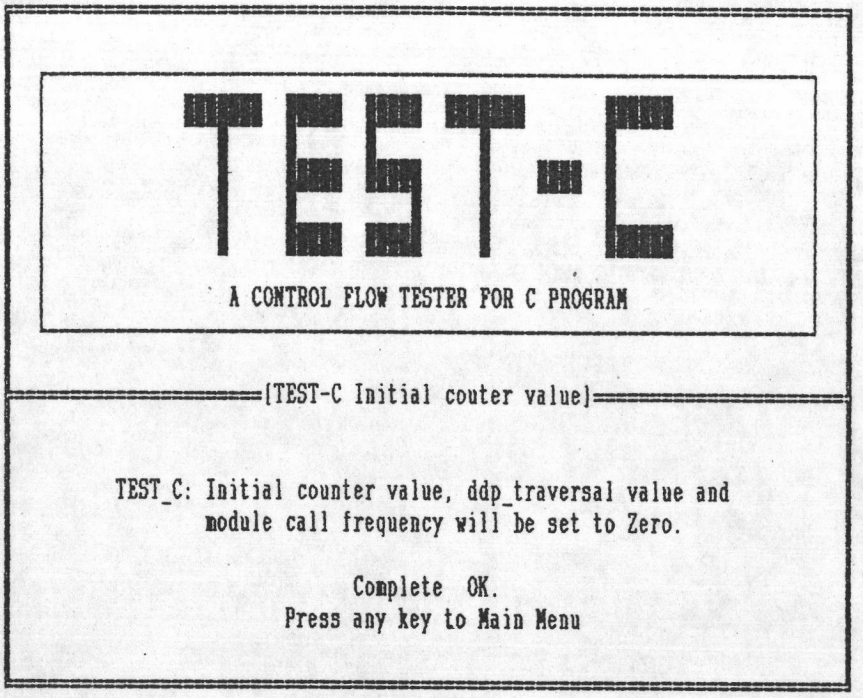
รูป 3.5.12 แสดงรูปจอภาพของ Complete ddp_path coverage report



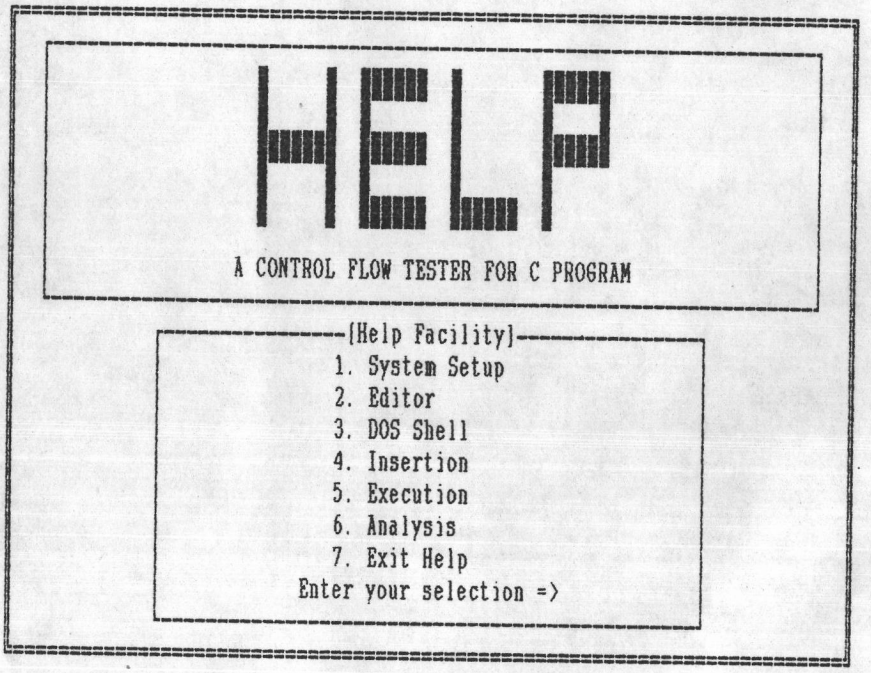
รูป 3.5.13 แสดงรูปจอภาพของ Untraversed dd-path coverage report



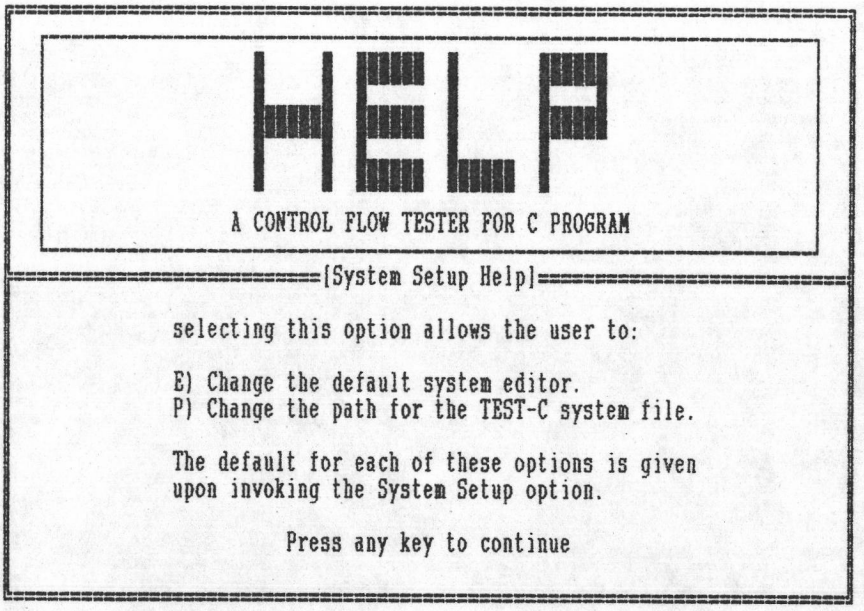
รูป 3.5.14 แสดงรูปจอภาพของ Module call frequency report



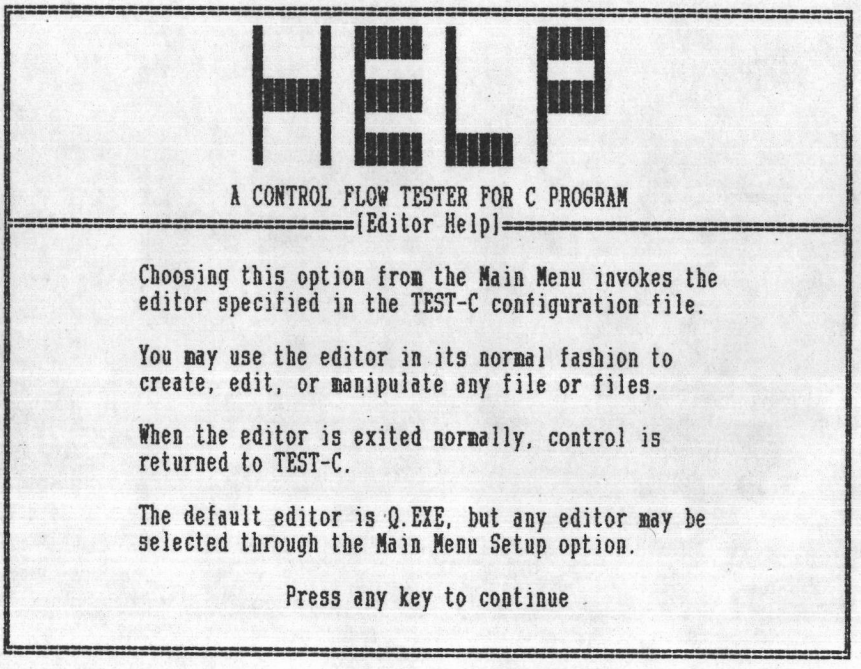
รูป 3.5.15 แสดงรูปจอภาพของ Clear counter



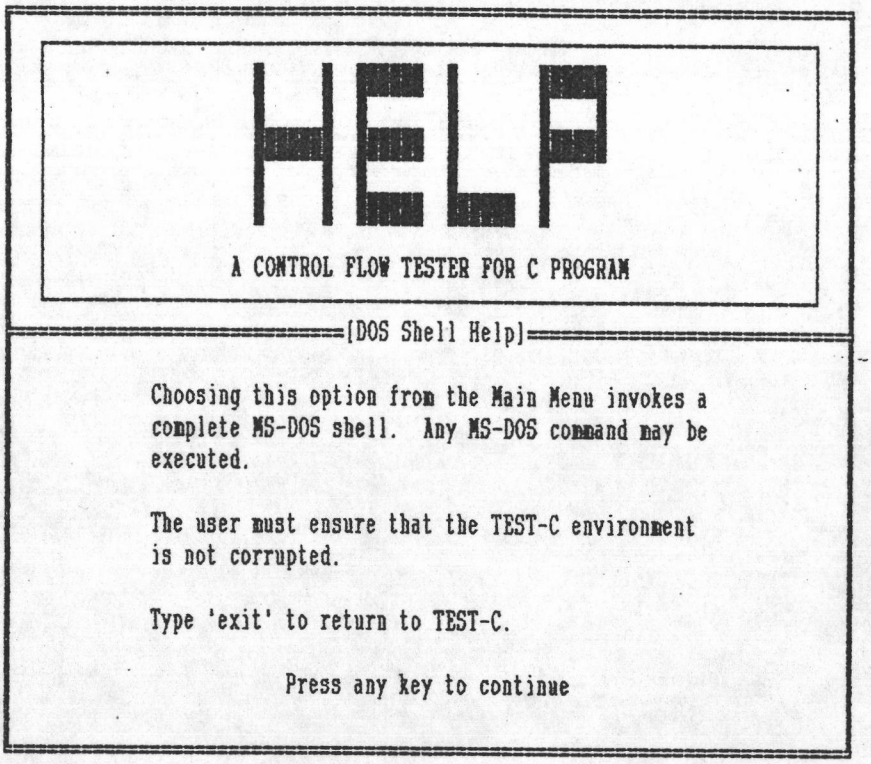
รูป 3.5.16 แสดงรูปจอภาพของ Help Menu



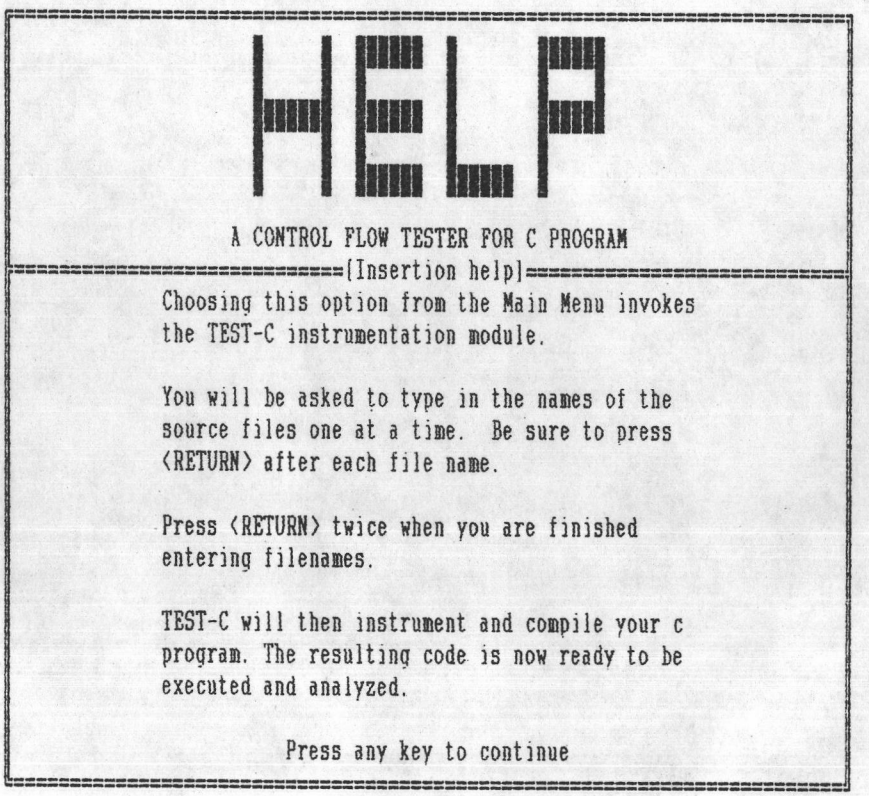
รูป 3.5.17 แสดงรูปจอภาพของ System Setup Help



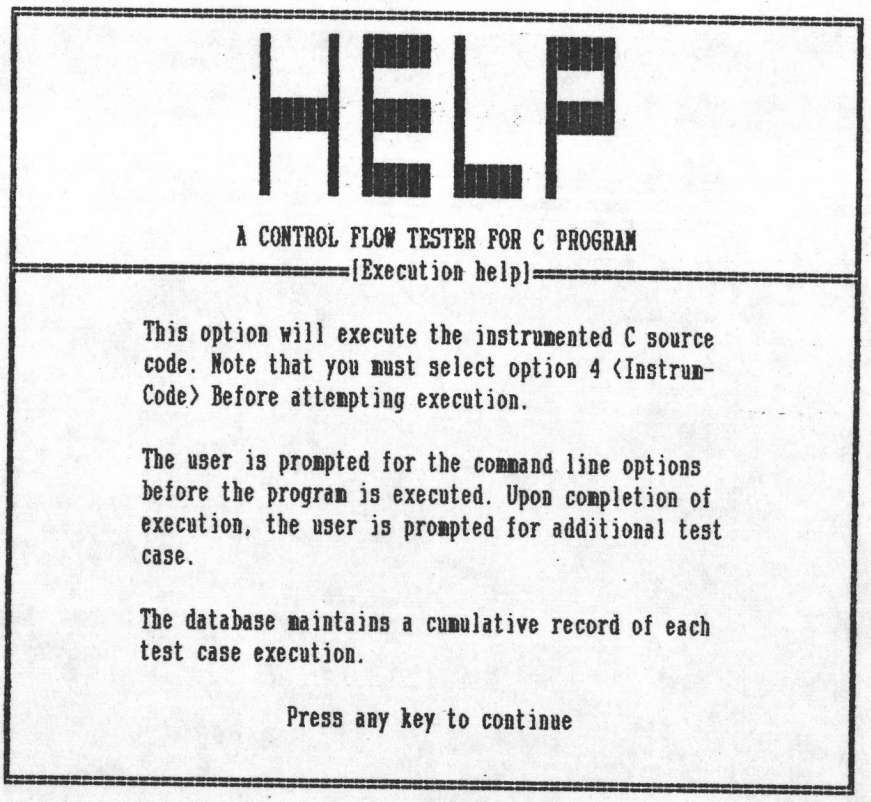
รูป 3.5.18 แสดงรูปจอภาพของ Editor Help



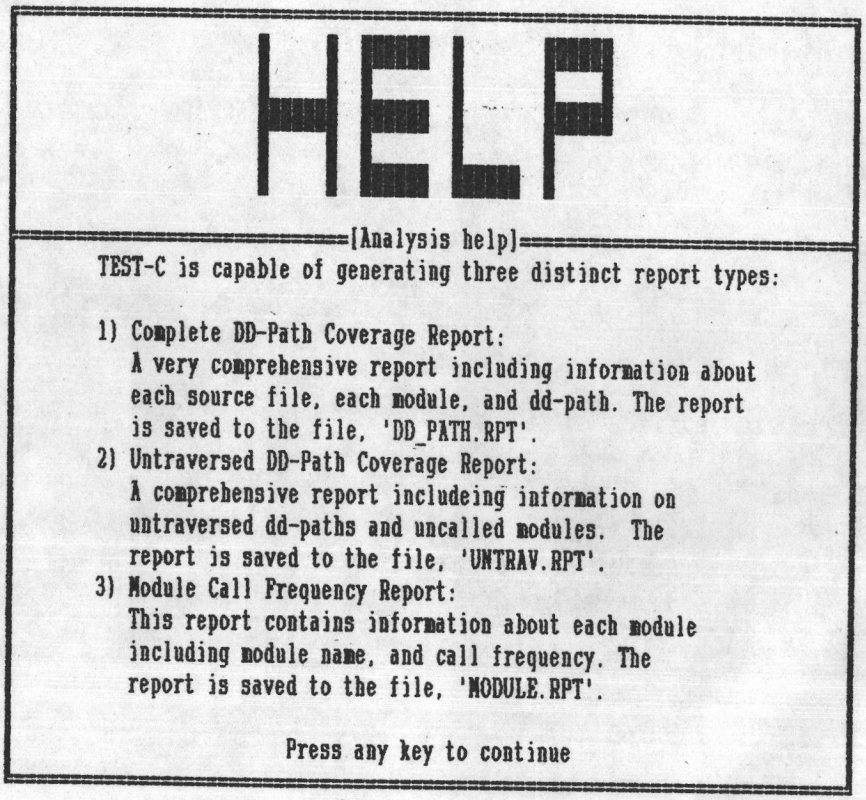
รูป 3.5.19 แสดงรูปจอภาพของ Dos Shell Help



รูป 3.5.20 แสดงรูปจอภาพของ Insertion Help



รูป 3.5.21 แสดงรูปจอภาพของ Execution Help



รูป 3.5.22 แสดงรูปจอภาพของ Analysis Help

3.6 ออกแบบผลลัพธ์ และรายงาน

รูปแบบรายงานผลลัพธ์ที่ได้ออกแบบไว้ดังแสดงได้ดังรูปที่ 3.6.1 ถึง 3.6.3 ดังต่อไปนี้

```
                    COMPLETE DD_PATH COVERAGE REPORT
                    --------------------------------

Number of files : 1
Total number of dd_path : 162
=============================================================================
TEST_C  TEST_C  TEST_C  TEST_C  TEST_C  TEST_C  TEST_C  TEST_C  TEST_C
=============================================================================

FILE : ftype.c
Total number of Module : 30
==========================================

MODULE NAME : main
------------------------------------------
Source code line number : 38
Total number of dd_path : 53
   dd_path              source code      traversal      percentage of
   keyword              line number       count         traversal total
-----------------------------------------------------------------------------
if                          42              0           0.00000
dd_path else condition      45              1           0.00035
if                          46              0           0.00000
dd_path else condition      49              1           0.00035
if                          53              0           0.00000
dd_path else condition      56              1           0.00035
if                          57              0           0.00000
dd_path else condition      60              1           0.00035
while                       64            646           0.22411
if                          66             23           0.00798
dd_path else condition      69            623           0.21613
if                          70             11           0.00382
dd_path else condition      73            612           0.21232
while                       74           2869           0.99532
if                          76             42           0.01457
dd_path else condition      80           2827           0.98075
if                          81             65           0.02255
dd_path else condition      84           2762           0.95820
if                          85            179           0.06210
if                          88             15           0.00520
```

รูป 3.6.1 ตัวอย่างรายงานการประมวลผลของคำสั่งควบคุมสายงานทั้งหมด

```
                    UNTRAVERSAL DD_PATH COVERAGE REPORT
                    ----------------------------------

Number of files : 1
Total number of dd_path : 162
=============================================================================
TEST_C  TEST_C  TEST_C  TEST_C  TEST_C  TEST_C  TEST_C  TEST_C  TEST_C
=============================================================================

FILE : ftype.c
Total Number of Module : 30
==========================================

MODULE NAME: main
------------------------------------------
Source code line number: 38
Total number of dd_path: 53
    dd_path              source code          traversal
    keyword              line number          count
-----------------------------------------------------------
if                          42                  0
if                          46                  0
if                          53                  0
if                          57                  0
case                        106                 0
dd_path else condition      172                 0

MODULE NAME: write_fp2
-----------------------------------------------
Source code line number: 187
Total number of dd_path: 2
    dd_path              source code          traversal
    keyword              line number          count
-----------------------------------------------------------
dd_path else condition      195                 0

MODULE NAME: skip_space
-----------------------------------------------
Source code line number: 199
Total number of dd_path: 2
    dd_path              source code          traversal
    keyword              line number          count
-----------------------------------------------------------
      -=*   No untraversal in this module   *=-
```

รูป 3.6.2 ตัวอย่างรายงานการของคำสั่งควบคุมสายงานที่ไม่ได้ประมวลผล

MODULE CALL FREQUENCY REPORT

Total number of files: 1

TEST_C TEST_C TEST_C TEST_C TEST_C TEST_C TEST_C TEST_C TEST_C

FILE : ftype.c

Total number of module: 30

module name	source code line number	call frequency	percentage
main	38	1	0.00611
write_fp2	187	812	4.95999
skip_space	199	3818	23.32173
isspace_line	209	646	3.94600
handle_space_line	220	23	0.14049
iscomment	228	2869	17.52489
handle_c_comment	236	42	0.25655
is_preprocess_keywords	283	623	3.80551
handle_preprocess_key	296	11	0.06719
is_declare_keywords	308	2827	17.26834
handle_declare_key	325	65	0.39704
is_statement_keywords	339	2762	16.87130
get_next_line	356	681	4.15979
handle_asm_statement	368	0	0.00000
handle_label_statement	377	22	0.13438
handle_jump_statement	406	81	0.49478
handle_statement_if	414	52	0.31763
handle_statement_switch	430	5	0.03054
handle_statement_while	445	14	0.08552
handle_statement_do	461	1	0.00611
handle_statement_for	472	3	0.01833
handle_statement_else	488	1	0.00611
handle_statement_sizeof	498	0	0.00000
handle_string_literal	506	24	0.14660
find_char	535	169	1.03231
handle_single_quote	572	7	0.04276
handle_parenthesis	584	104	0.63527
check_module	628	515	3.14581
write_module	664	30	0.18325
write_dd_path	673	163	0.99566

รูป 3.6.3 ตัวอย่างรายงานความถี่ในการเรียกใช้โมดุล

3.7 ข้อกำหนดของระบบ

ข้อกำหนดของระบบการทดสอบสายงานควมคุมของโปรแกรมภาษาซี (TEST_C) นั้น ผู้ใช้งานจะต้องเป็นผู้ดูแล และจัดการเกี่ยวกับข้อกำหนดของระบบ ซึ่งมีดังต่อไปนี้

3.7.1 ไวยากรณ์ (Syntax)

โปรแกรมรหัสต้นฉบับภาษาซี ที่จะทำการทดสอบนั้นจะต้องมีไวยากรณ์ถูกต้องอยู่แล้ว โดยเฉพาะอย่างยิ่งการประมวลโปรแกรมนั้นจะต้องสามารถที่จะประมวลผลได้เป็นปกติ ซึ่งผู้ใช้จะเป็นผู้จัดการเองรวบถึงการทำงานต่างๆ ของระบบด้วย

3.7.2 ข้อผิดพลาดต่างๆของระบบปฏิบัติการ (DOS Errors)

ความผิดพลาดของระบบปฏิบัติการ (DOS) ที่เกิดขึ้นในระหว่างการใช้งาน TEST_C จะไม่ควบคุมจัดการโดยตรง ในกรณีเช่นนี้ผู้ใช้จะต้องระวังและจัดการข้อผิดพลาดที่เกิดจากระบบปฏิบัติการเอาเอง

3.7.3 ตัวแปลชุดคำสั่ง (Compiler)

ตัวแปลชุดคำสั่งในที่นี้จะใช้ เทอร์โบซี รุ่น 2.0 เป็นตัวแปลชุดคำสั่ง และโปรแกรมรหัสต้นฉบับภาษาซีที่จะใช้ทดสอบในที่นี้ก็เป็น โปรแกรมรหัสต้นฉบับภาษาซีสำหรับตัวแปลชุดคำสั่งภาษาซีนี้ด้วยเช่นกัน

3.7.4 รหัสต้นฉบับภาษาซี

รหัสต้นฉบับภาษาซี ที่จะใช้ทดสอบนั้น จะต้องมีทางเข้า และทางออกที่โมดุล main เท่านั้น และการจบโปรแกรมใน main ให้จบด้วยคำสั่ง return; หรือจบตามปกติของโปรแกรม และคำสั่งที่เกี่ยวกับคำสั่งควบคุมสายงานต้องใส่วงเล็บปีกกาให้ครบถ้วน