Unit 4. Handling Exceptions in CICS

Objectives

Exception Handling Commands such as:

HANDLE CONDITION

IGNORE CONDITION

NOHANDLE and RESP OPTIONS

PUSH and POP

ADDRESS and ASSIGN

GETMAIN and FREEMAIN

Figure: 4-1. Objectives

Notes :

Exception Conditions

- **A CICS command that cannot get executed will result in a exception.**

- **It may occur at any time during the execution of a command and unless otherwise specified in the application program.**

- **CICS issues a Abnormal Termination Code (ABEND Code) if the task terminates abnormally.**

Figure: 4-2. Exception Conditions.

Notes :

Exception is a run time error.
Exceptions can be handled by some CICS commands such as HANDLE,NOHANDLE, IGNORE And RESP.

HANDLE CONDITION Command

- **This command allows an application program to check for exceptions that may occur when CICS commands are executed.**
- **It specifies which conditions are to be checked as well as the names of the exception routines to be executed while encountering the condition.**
- **It is used to transfer control to the procedure label specified if the exception condition specified occurs.**
- **Once the HANDLE CONDITION command request is made it remains active until the end of the program or another HANDLE CONDITION command overrides it.**

Figure: 4-3. HANDLE CONDITION Command.

Notes :

Branch is a user provided paragraph name where a user written generalized routine can respond to a category of conditions.
Handle Command indicate where to branch when a particular exception occurs.

HANDLE CONDITION Command Syntax

```
EXEC CICS HANDLE CONDITION
          Condition(label)
          [condition(label)]
          [ERROR(label)]
END-EXEC.
```

Condition **Represents exception condition**

Label **The name of the paragraph or routine to which the control will be passed when specified condition occurs.**

Figure: 4-4. HANDLE CONDITION Command Syntax.

Notes :

```
EXEC CICS HANDLE CONDITION LENGERR(LONGMSG)
EXEC CICS RECEIVE INTO(INMSG) LENGTH(MSGLEN) ERROR(GENERR)…
……
……
EXEC CICS HANDLE CONDITION LENGERR(LONGREC)
EXEC CICS READ DATASET('MASTER') RIDFLD(RECKEY) INTO(RECAREA)
LENGTH(RECLEN)..
……
……
```

IGNORE CONDITION Command

- **IGNORE CONDITION requests CICS to ignore condition occurrences of specified exception conditions.**
- **When a specified condition occurs CICS returns control to the application program at the instruction following the command that caused the condition to occur.**
- **The request by the IGNORE CONDITION command is valid until the next HANDLE CONDITION command for the same condition.**

Figure: 4-5. IGNORE CONDITION Command.

Notes :

Ignore condition command resets the branching mechanism for specified exceptions. Ignore condition causes CICS to return to the next statement in the application program if the named condition occurs.

IGNORE CONDITION Command Syntax

EXEC CICS IGNORE CONDITION
Condition
[condition]
END-EXEC.

Figure: 4-6. IGNORE CONDITION Command Syntax.

Notes :

Example :

```
EXEC CICS IGNORE CONDITION MAPFAIL
EXEC CICS RECEIVE MAP('MAINTEC') MAPSET('MAINTEC')……..
……………
……..
……..
```

In the above example if a MAPFAIL condition occurs on the RECEIVE MAP command control is returned to the instruction immediately following the RECEIVE MAP command.

NOHANDLE option

Example :

```
EXEC CICS READ DATASET('MAINTEC')
     LENGTH(RECLEN)
     INTO(RECAREA)
     RIDFLD(RECKEY)
     NOHANDLE
END-EXEC.
```

Figure: 4-7. NOHANDLE option.

Notes :

The NOHANDLE option in this command requests CICS to ignore any exception condition that occurs when the command is executed. If an exception condition occurs, CICS returns control to the application program at the instruction immediately following the READ command. The program can then analyze the command response code in the EIB to determine the results of the command.

RESP Option

- **RESP option is similar to the NOHANDLE option.**
- **CICS returns control to the application program at the instruction immediately following the command.**
- **When RESP is coded in a command, CICS returns a response code to the application program a predefined work storage field.**
- **The response code field returned can be used by the application program, using CICS supplied values, to determine which exception condition has occurred.**

Figure: 4-8. RESP option.

Notes :

RESP technique requires the application program to test the execution condition following each command.

RESP Option Example

```
EXEC CICS RECEIVE INTO(INMSG)
        LENGTH(MSGLEN)
        RESP(RECVRSP)
END-EXEC.


IF RECVRSP=DFHRESP (LENGERR) THEN
…….
……..
ELSE
    IF RECVRSP NOT = DFHRESP (NORMAL) THEN
    ……
    ……
    ELSE
    …….
    …….
```

Figure: 4-9. RESP option Example.

Notes :

In the above example , the RECVRSP is the field specified following the RESP option on the preceding RECEIVE command. The program compares the contents of RECVRSP to a list of CICS supplied response code values, referred to as DFHRESP(condition).

These CICS supplied values are automatically included in the application program, no definition is to be coded by the programmer.

The response field is to be defined in the Working storage section, should be a full word binary field (PIC S9(8) COMP).

PUSH and POP Commands

- **PUSH and POP Commands are used to suspend and reactivate respectively, all HANDLE CONDITION requests that are currently in effect.**
- **These commands are useful when performing a subroutine embedded in a main program.**
- **A called routine can use the PUSH command to suspend the existing HANDLE CONDITION.**
- **A POP command can be used before returning control to the caller to reactivate the HANDLE CONDITION.**

Figure: 4-10. PUSH and POP Commands.

Notes :

PUSH and POP Command Syntax

PUSH Command :

```
EXEC CICS PUSH HANDLE
END-EXEC.
```

POP Command :

```
EXEC CICS POP HANDLE
END-EXEC.
```

Figure: 4-11. PUSH and POP Command Syntax.

Notes :

Example :

MAIN PROGRAM.

```
…….
…….
EXEC CICS HANDLE CONDITION
…….
END-EXEC.
……
PERFORM SUBROUTINE-A
…….
EXEC CICS HANDLE CONDITION
…….
END-EXEC.
…….
…….
```

SUBROUTINE-A

```
EXEC CICS PUSH HANDLE
END-EXEC.
…….
…….
EXEC CICS HANDLE CONDITION
…….
END-EXEC.
…….
…….
EXEC CICS POP HANDLE
END-EXEC.
SUBROUTINE-A-EXIT.
EXIT.
```

ADDRESS and ASSIGN Commands

ADDRESS Command :

```
EXEC CICS ADDRESS
        option (ADDRESS of data-area)
        [option(ADDRESS of data-area)]
END-EXEC.
```

ASSIGN Command:

```
EXEC CICS ASSIGN
        option(data-area)
        [option(data-area)]
END-EXEC.
```

Figure: 4-10. ADDRESS and ASSIGN Commands.

Notes :

The ADDRESS command is used to access information in the CICS system areas.

The ASSIGN command is used to access the system value outside of the application program.

Commonly used options are :

CWALENG	To access the length of the CWA.
TCTUALENG	To access the length of the TCTUA.
TWALENG	To access the length of TWA.
USERID	To access the user-id(8 character field).
ABCODE	To access the ABEND code(4 character field).

GETMAIN and FREEMAIN Commands

GETMAIN Command :

```
EXEC CICS GETMAIN
        SET(ADDRESS of data-area)
        LENGTH(data-value) | FLENGTH(data-value)
        [INITIMG(data-value)]
        [NOSUSPEND]
END-EXEC.
```

FREEMAIN Command :

```
EXEC CICS FREEMAIN
        DATA(data-area)
END-EXEC.
```

Figure: 3-12. GETMAIN and FREEMAIN Commands.

Notes :

Using GETMAIN command a program can acquire a specified amount of virtual storage.

The storage acquired through GETMAIN should be made free as soon as its requirement is over, this is achieved by issuing a FREEMAIN command. If the FREEMAIN command is not used the acquired storage will be released only at the termination of the task. You cannot issue a FREEMAIN command to release an area that was not acquired using a GETMAIN command.