

## Chapter 16

# Improving Program Productivity Using The COPY and CALL Statements

# Chapter Objectives

- Explain the purpose of the COPY statement and describe its use
- Explain the purpose of the CALL statement and describe its use
- Demonstrate how the CALL statement is used for dynamic and static program calls

# COPY Statement

- To copy prewritten COBOL entries stored in a library into a program
- Often used to copy
  - FD, 01 entries that define and describe files and records
  - Tables
  - Standard modules used in PROCEDURE DIVISION

# Benefits of COPYing Entries

- Saves programmer coding, debugging time by using code already written
- Promotes program standardization - all entries copied from library are identical
- Reduces time needed to modify code - changes made once to library copy
- Library entries can be well-documented so meaningful to all users

# COPY Statement

COPY source-member { OF  
IN } library-source file.

- Use in ENVIRONMENT, DATA, or PROCEDURE DIVISION
- The source member is the name of the source member to be copied
- To qualify the source file, a hyphen is required between library and source file.

# COPY Statement Example

- Suppose a source member called Customer, stored in library COBLIB contains following entries:

01 Customer-Rec.

05 Cust-No Pic X(5).

05 Cust-Name Pic X(20).

05 Cust-Address Pic X(30).

05 Cust-Bal-Due Pic 9(4)V99.

# COPY Statement Example

- Code COPY statement at point in program where entries should appear  
Data Division.  
File Section.  
FD CustFile.  
Copy Customer.
- In source listing, copied lines are marked with a + or other designator

# Source Listing with COPY

```
...      ...  
11      FD      CustFile.  
12      Copy Customer IN COBLIB-QCBLSRC.  
13+     01      Customer-Rec.  
14+           05      Cust-No           Pic X(5).  
15+           05      Cust-Name        Pic X(20).  
16+           05      Cust-Address     Pic X(30).  
17+           05      Cust-Bal-Due     Pic 9(4)V99.
```

# CALL Statement

- Used in PROCEDURE DIVISION to CALL or reference independent subprograms stored in library
- Calling program - main program that calls or references subprogram
- Called program - subprogram stored in separate file that is linked and executed within main program

# Called Program (Subprogram)

- Compiled, debugged, stored in library to be called when needed
- Typical subprograms include:
  - Edit routines
  - Error control checks
  - Standard calculations
  - Summary, total printing

# Advantages of Subprograms

- Avoids duplication of effort
- Improves programmer productivity
- Provides greater flexibility
- Changes to called program can be made without modifying calling program
- Results in greater standardization

# CALL vs COPY

- COPY brings entries into program as is
- CALL causes an entire program (the called program) to be executed
  - Data passed from calling to called program
  - Entire called program (subprogram) executed
  - Data passed back from called program
  - Control returns to calling program

# CALL vs COPY

- Typically, COPY ENVIROMENT, DATA DIVISION entries into source program
- CALL subprograms in PROCEDURE DIVISION rather than COPY them

# CALL Statement

CALL literal-1  
[USING identifier-1 ...]

- To call subprogram into main program
- literal-1 is PROGRAM-ID name in called program
- Enclose in quotes

# Calling and Called Programs

Identification Division.

...

Procedure Division.

...

Call 'Sub1'

1

2

...

Call 'Sub2'

3

...

4

Identification Division.

Program-ID. Sub1.

...

Procedure Division.

Exit Program.

Identification Division.

Program-ID. Sub2.

...

Procedure Division.

Exit Program.

# Calling and Called Programs

1. Subprogram Sub1 called and executed in its entirety
2. Control returns to calling program
3. Subprogram Sub2 called and executed in its entirety
4. Control returns to calling program

# CALL ... USING

- USING clause required if data passed between calling and called program
- Fields passed are called parameters
- Parameter passed in several ways:
  - Value of parameter may be passed to and used by subprogram but not changed
  - Value for parameter may be calculated by instructions in subprogram and passed back to main program

# CALL ... USING Example

- Suppose a main program
  - Reads in employee records
  - Produces report showing employee salary, Social Security tax and Medicaid tax (taxes referred to as FICA taxes)
- Subprogram is to calculate amount of each tax based on the salary field
  - Assume PROGRAM-ID is FICAProg

# CALL ... USING Example

- Parameters needed
  - Salary (WS-Ann-Sal)
    - Passed to subprogram, used to calculate taxes but not changed by subprogram
  - Tax fields Soc-Sec and Med-Tax
    - Subprogram calculates values for these parameters and passes them back to main program

# Calling Program Requirements

- CALL statement with:
  - Literal same as PROGRAM-ID of called program
  - List of parameters in USING clause

Call 'FICAProg'

Using WS-Ann-Sal, Soc-Sec, Med-Tax

# Called Program Requirements

- PROGRAM-ID identifier is literal used in CALL in main program
- LINKAGE SECTION must be defined in DATA DIVISION
  - Follows FILE and WORKING-STORAGE SECTIONS
  - Describes all items passed between main and subprogram
  - Format same as WORKING-STORAGE

# Called Program Requirements

- PROCEDURE DIVISION header must include USING clause
  - Lists all parameters or fields defined in LINKAGE SECTION
- EXIT PROGRAM
  - Last executed statement in called program
  - Returns control back to calling program

# Subprogram Example

Identification Division.

Program-ID. FICAProg.

Data Division.

Linkage Section.

01      WS-Ann-Sal      Pic 9(6).

01      Soc-Sec      Pic 9(4)V99.

01      Med-Tax      Pic 9(5)V99.

# Subprogram - Procedure Division

## Procedure Division

Using WS-Ann-Sal, Soc-Sec, Med-Tax.

If WS-Ann-Sal <= 84900

    Compute Soc-Sec = .062 \* WS-Ann-Sal

Else

    Compute Soc-Sec = .062 \* 84900

End-If

Compute Med-Tax = .0145 \* WS-Ann-Sal

Exit Program.

# Subprogram Example

- When FICAProg is called
  - Value of WS-Ann-Sal passed to subprogram
  - Value of Soc-Sec and Med-Tax undefined
- When FICAProg finished
  - Values calculated for Soc-Sec and Med-Tax passed back to main program

# Parameter Correspondence

- Parameters passed in sequence
  - First parameter in CALL ... USING passed to first parameter in PROCEDURE DIVISION USING clause and so on
- Number of parameters in both USING clause should be the same
- PIC clauses for corresponding parameters must be same

# Parameter Correspondence

- Data-names of corresponding parameters may be same or different
- Parameters paired by sequence, not by data-name
- List parameters in same order in both USING clauses

# Parameter Correspondence

- Assume USING clauses are:  
Call 'FICAProg'  
Using Soc-Sec, Med-Tax, WS-Ann-Sal  
Procedure Division  
Using WS-Ann-Sal, Soc-Sec, Med-Tax.
  - Soc-Sec passed as value for WS-Ann-Sal
  - Calculations in subprogram will be incorrect, values passed back incorrect

# Dynamic Program Calls

- A dynamic program CALL transfers control from a calling program to a called program that has been compiled into a separate program object(\*PGM)
- Both programs are compiled into separate program objects(\*PGM)
- The called program is loaded and executed only when it is called.
- The LINKAGE TYPE PROGRAM clause is used for dynamic program calls

# Static Program Calls

- The Integrated Language Environment (ILE) supports dynamic and static program calls
- A static procedure CALL transfers control from a calling program to a subprogram that is compiled with the calling program before run time occurs
- This means that there is only one executable program object (\*PGM) that contains the executable objects from both programs
- The LINKAGE TYPE PROCEDURE clause is used for static program calls

# Create Module for Static Bound Programs

- Each program must be compiled into module(\*MODULE) using the CRTCBLMOD (Create COBOL Module) command
- The resulting \*MODULE object is non-executable, intermediate representation of the source member

# Create Program for Static Bound Programs

- Once the individual programs are compiled into modules, they are bound into an executable program object (\*PGM) using the CRTPGM (Create Program) command

# Chapter Summary

## COPY Statement

- To copy entries stored in a library to a user program
- Entries for ENVIRONMENT, DATA, and PROCEDURE DIVISIONs may be copied

# Chapter Summary

COPY statement most often used for

- Copying standard file and record description entries
- Modules used in PROCEDURE DIVISION

# Chapter Summary

## CALL Statement

- To call or reference entire programs stored in a library
- Calling program is user program
- Called program (subprogram) is program accessed from library

# Chapter Summary

To pass data between calling program and called program

- CALL statement must include USING clause listing name of fields being passed
- PROCEDURE DIVISION statement of called program must also have USING clause
- Called program must have LINKAGE SECTION defining fields passed
- Identifiers in calling and called program may be same or different