# PERFORM USAGE CONVENTION – DEVELOPER TIP

USE THRU STAMENT AS BELOW

|  |
| --- |
| PERFORM 3600A-PROCESS-BATCH-UOW  THRU 3600A-EXIT |

# PARAGRAPH STARTING CONVENTION – DEVELOPER TIP

LEAVE ONE SPACE INBETWEEN OLD PARA ENDING AND NEW PARA LIKE BELOW

# PARAGRAPH ENDING CONVENTION – DEVELOPER TIP

FOR BEST PRACTICES, USE

1. ‘4000M-EXIT.’ IN COLUMN 8
2. ‘EXIT.’ IN COLUMN 12
3. ‘EJECT’ IN COLUMN 12

**PCII STANDARDS BELOW FOR REFERENCE:**

|  |
| --- |
| 3600A-PROCESS-BATCH-UOW.  INITIALIZE DETAIL-LINE    PERFORM 3620M-GET-LOCAL-TIMESTAMP    IF INTERNAL-RESTART  CONTINUE  ELSE    PERFORM 4000M-PROCESS-RECORD    END-IF  .  3600A-EXIT.  EXIT.  EJECT |

# PARAGRAPH NAMING CONVENTION – DEVELOPER TIP

In PCII Application, below is the paragraph naming convention

9990H-ABNORMAL-END-OF-JOB => ‘H-‘ INDICATES SHELL SPECIFIC CODE

1200A-APPL-SPECIFIC-INIT => ’A-‘ INDICATES APPLICATION SPECIFIC CODE

5410M-OPEN-INPUT-FILE =>’M-‘ INDICATES DEVELOPER MANUALLY ADDED PARA

PCII STANDARDS:

3000 SERIES - 3610M - TIMESTAMP EXTRACTION

4000 SERIES - 4990M - DEVELOPER PARAS (INSERT CJ ROW IN E29)

5000 SERIES - 5950M - FILE PROCESSINGS

8000 SERIES - 8300A - CHECKPOINT DATE

PARAGRAPH DECLARATION - INCREMENT TO BE FOLLOWED: (4000 TO 5000)

INCREMENT BY 1 FROM 4000 TO 5000 = 1000 PARA DECLARATION POSSIBLE

INCREMENT BY 2 FROM 4000 TO 5000 = 500 PARA DECLARATION POSSIBLE

INCREMENT BY 5 FROM 4000 TO 5000 = 200 PARA DECLARATION POSSIBLE

INCREMENT BY 10 FROM 4000 TO 5000 = 100 PARA DECLARATION POSSIBLE (SEEMS BEST)

INCREMENT BY 20 FROM 4000 TO 5000 = 50 PARA DECLARATION POSSIBLE

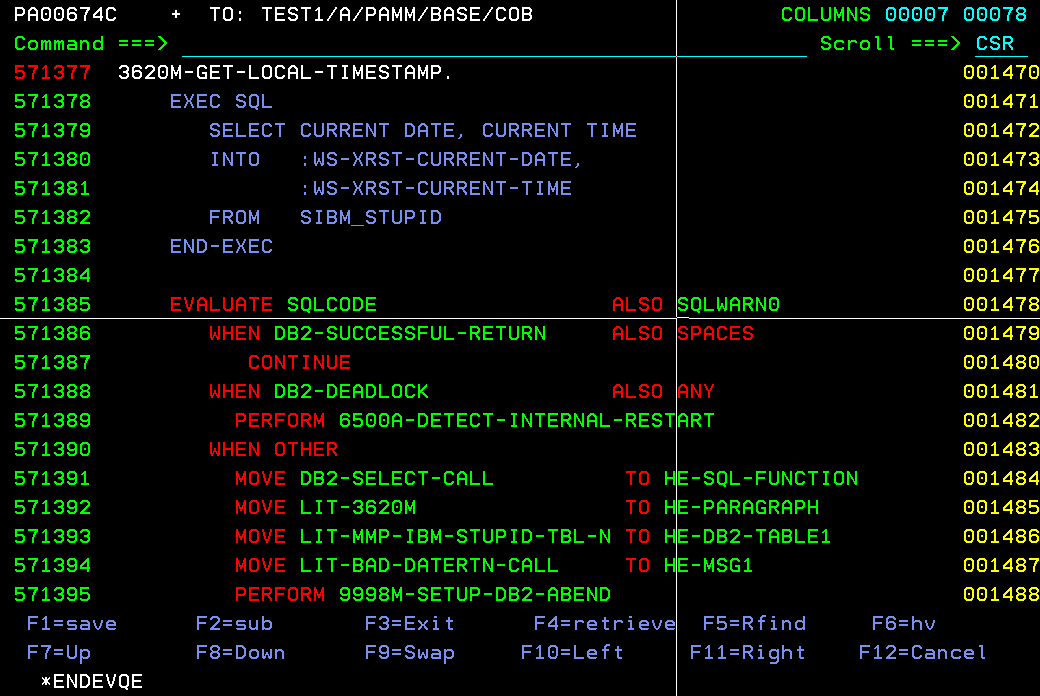
INCREMENT BY 50 FROM 4000 TO 5000 = 20 PARA DECLARATION POSSIBLE

INCREMENT BY 100 FROM 4000 TO 5000 = 10 PARA DECLARATION POSSIBLE

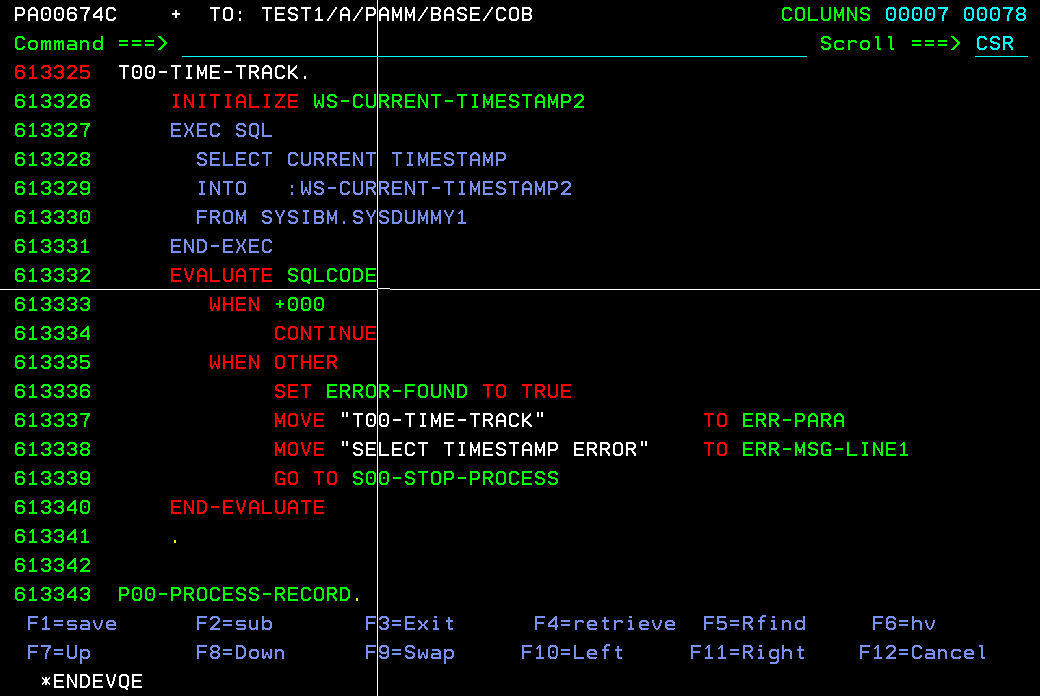
# SQLCODE NAMING CONVENTION – DEVELOPER TIP

AFTER EXECUTING A QUERY, CHECK THE SQLCODE ALONG WITH SQLWARNING, DO NOT CHECK FOR SQLCODE ALONE.

CORRECT METHOD:



WRONG METHOD:



# comp and comp3 variables explain

Sure! Let's break down **COMP** and **COMP-3** variables in COBOL:

### COMP (Computational)

* **Usage**: The **COMP** clause is used to define binary data items. It is often used for efficient arithmetic operations.
* **Storage**: The storage size depends on the number of digits specified in the **PICTURE** clause:
  + 1 to 4 digits: 2 bytes (halfword)
  + 5 to 9 digits: 4 bytes (fullword)
  + 10 to 18 digits: 8 bytes (doubleword)
* **Example**:

01 WS-BINARY-VAR PIC S9(4) USAGE IS COMP.

This defines a signed binary variable that can store up to 4 digits.

### COMP-3 (Packed Decimal)

* **Usage**: The **COMP-3** clause, also known as packed decimal, is used to store decimal numbers in a compact binary-coded decimal (BCD) format. It is efficient for arithmetic operations involving decimal numbers.
* **Storage**: Each digit is stored in 4 bits (half a byte), and the sign is stored in the last 4 bits. The storage size is calculated as:
  + For an odd number of digits: ((n + 1) / 2) bytes
  + For an even number of digits: (n / 2) bytes
* **Example**:

01 WS-PACKED-VAR PIC S9(5) USAGE IS COMP-3.

This defines a signed packed decimal variable that can store up to 5 digits, using 3 bytes of storage.

### Key Differences

* **COMP** is used for binary data and is efficient for integer arithmetic.
* **COMP-3** is used for packed decimal data and is efficient for decimal arithmetic.

Would you like more examples or further details on how to use these variables in COBOL?

[[1]](https://www.mainframestechhelp.com/tutorials/cobol/comp-3.htm): [Mainframes Tech Help - COMP-3](https://www.mainframestechhelp.com/tutorials/cobol/comp-3.htm) [[2]](https://ibmmainframes.com/about393.html): [IBM Mainframe Community - COMP, COMP1, COMP2 & COMP3](https://ibmmainframes.com/about393.html)

**References**

[1] [COBOL COMP-3 | Packed-decimal - www.www.mainframestechhelp.com](https://www.mainframestechhelp.com/tutorials/cobol/comp-3.htm)

[2] [COMP, COMP1, COMP2 & COMP3 in COBOL - IBM Mainframe Community](https://ibmmainframes.com/about393.html)

# what is diff b/w integer and decimal ? why comp and comp3 usage differentiates it?

### Difference Between Integer and Decimal

**Integer**:

* **Definition**: An integer is a whole number without any fractional or decimal part. It can be positive, negative, or zero.
* **Usage**: Integers are used for counting and discrete values.
* **Example**: 5, -3, 0

**Decimal**:

* **Definition**: A decimal number includes a fractional part, represented with a decimal point. It can also be positive or negative.
* **Usage**: Decimals are used for precise values, especially in financial calculations.
* **Example**: 5.75, -3.14, 0.0

### COMP vs. COMP-3 in COBOL

**COMP (Computational)**:

* **Data Type**: Binary
* **Usage**: Efficient for integer arithmetic operations.
* **Storage**: The storage size depends on the number of digits:
  + 1 to 4 digits: 2 bytes
  + 5 to 9 digits: 4 bytes
  + 10 to 18 digits: 8 bytes
* **Example**:

01 WS-BINARY-VAR PIC S9(4) USAGE IS COMP.

**COMP-3 (Packed Decimal)**:

* **Data Type**: Packed Decimal (Binary Coded Decimal)
* **Usage**: Efficient for decimal arithmetic operations.
* **Storage**: Each digit is stored in 4 bits, with the sign in the last 4 bits. The storage size is:
  + For an odd number of digits: ((n + 1) / 2) bytes
  + For an even number of digits: (n / 2) bytes
* **Example**:

01 WS-PACKED-VAR PIC S9(5) USAGE IS COMP-3.

### Why Differentiate COMP and COMP-3?

* **Efficiency**: **COMP** is more efficient for integer operations because it uses binary representation, which is faster for the CPU to process.
* **Precision**: **COMP-3** is used for decimal operations where precision is crucial, such as financial calculations. It stores numbers in a compact format, reducing storage space and maintaining accuracy.

Would you like more examples or further details on using these data types in COBOL?

[[1]](https://www.ibm.com/docs/en/cobol-zos/6.3?topic=arithmetic-formats-numeric-data): [IBM COBOL Numeric Data Formats](https://www.ibm.com/docs/en/cobol-zos/6.3?topic=arithmetic-formats-numeric-data) [[2]](https://www.ibm.com/docs/en/rbd/9.5.1?topic=types-numeric): [Mainframes Tech Help - COMP-3](https://www.mainframestechhelp.com/tutorials/cobol/comp-3.htm)

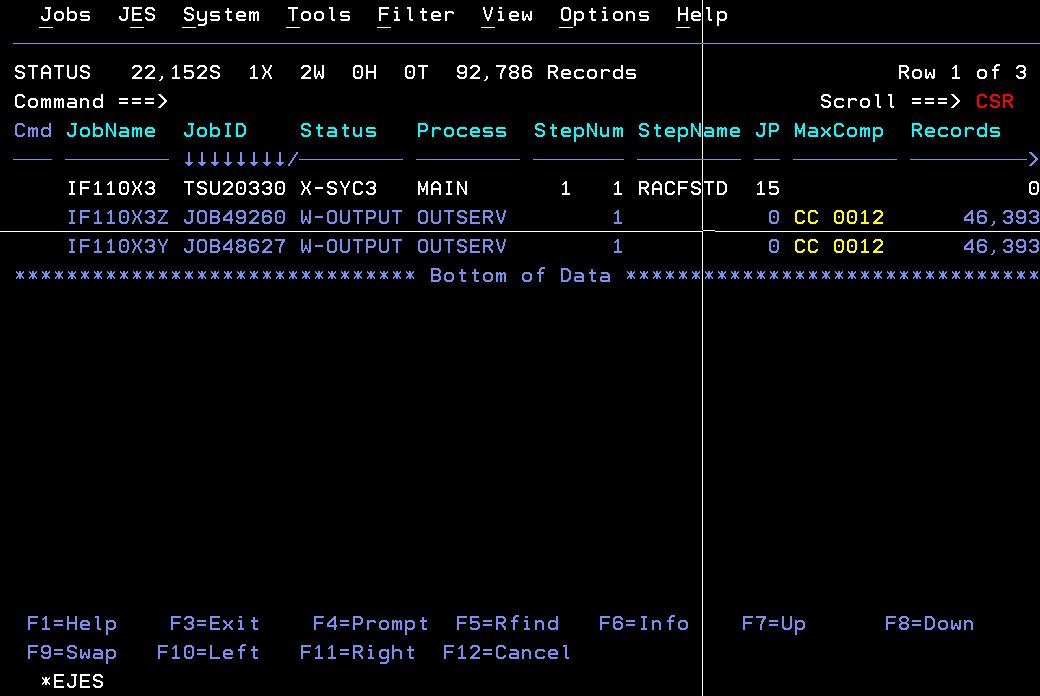
**References**

[1] [Formats for numeric data - IBM](https://www.ibm.com/docs/en/cobol-zos/6.3?topic=arithmetic-formats-numeric-data)

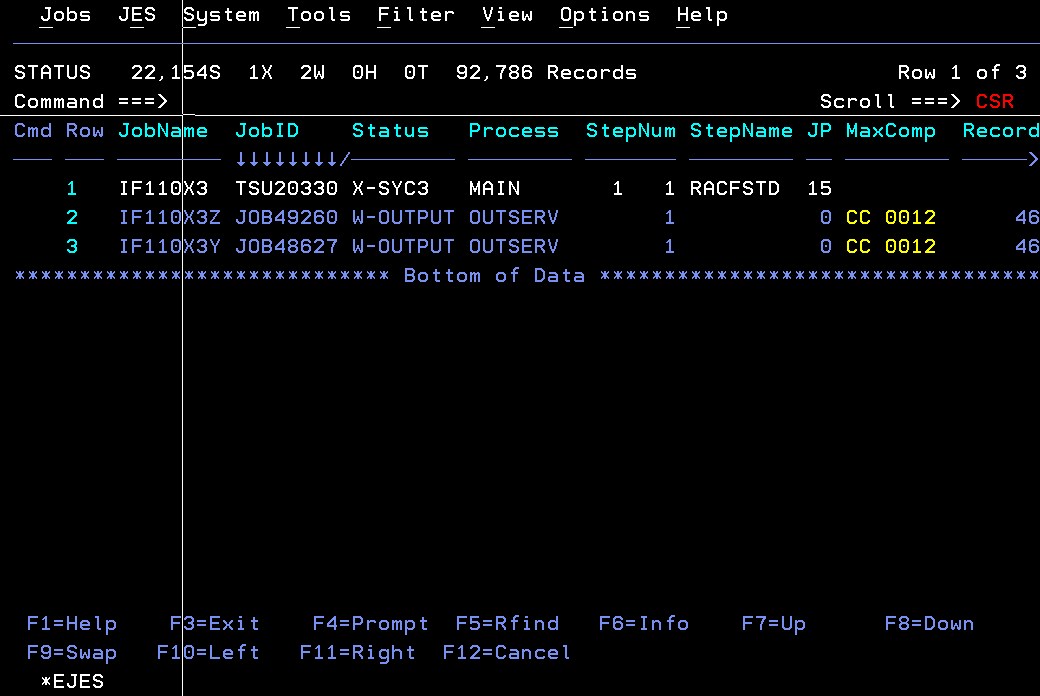
[2] [Numeric types - IBM](https://www.ibm.com/docs/en/rbd/9.5.1?topic=types-numeric)

# HOW TO ORGANIZE SPOOL/EJES/JES2/JES3 FOR THE JOBS (2 TECHNIQUES)

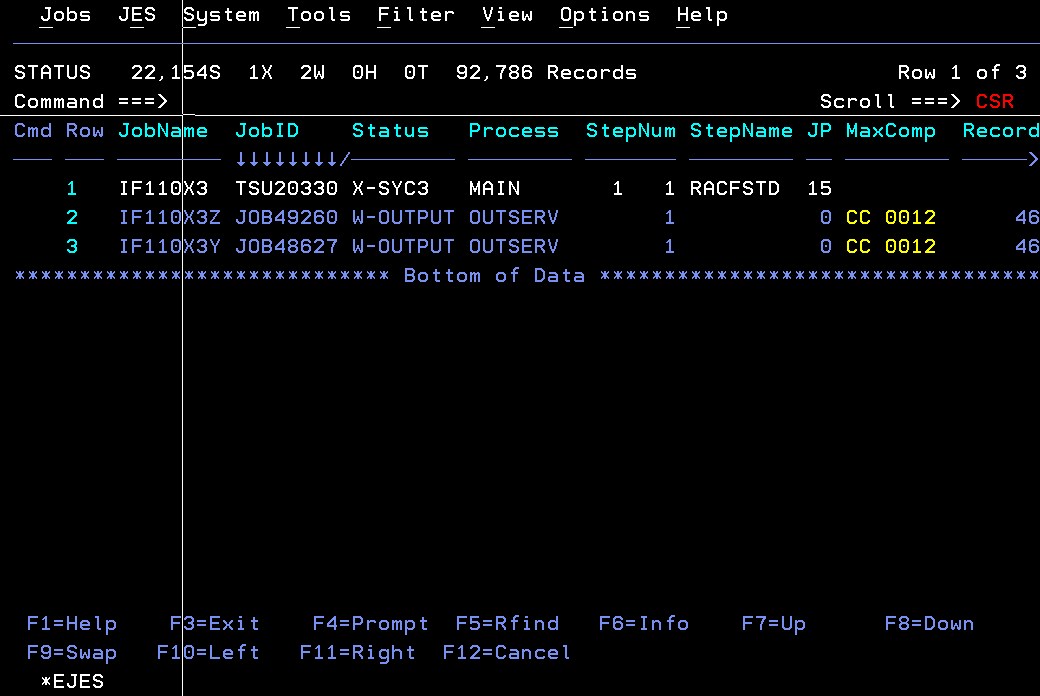
1. ROW COMMAND - BEFORE SHOWING ROW NUMBER (DEFAULT/ROW OFF)



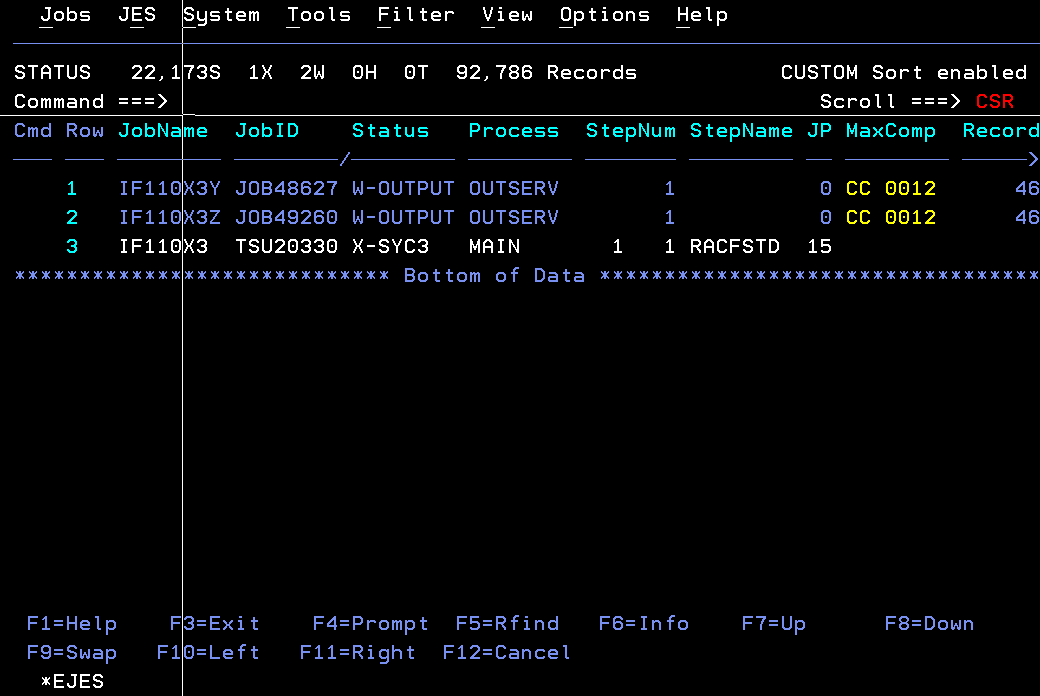
‘ROW ON’



1. JOB SORTING - BEFORE SORTING BY QUEUE DATE & QUEUE TIME (SORT COMMAND)



SORT BY QDate & QTime ascending(command= ‘SORT QDATE A QTIME A’)



# END OF DOCUMENT