Excel Output Command file [.EFX] Instructions

Overview

Excel Output Command files [*.EFX] are used to generate Excel files. Unlike the normal Excel output option, the command files allow only specific fields or expressions to be exported. There is also some limited capability to format the output by inserting titles and/or column captions and setting properties such as column width, bold, word wrap, etc. If pcMRP's proutput.prg detects an .efx with the same name as the report being generated, excelout.prg parses through the .efx file to generate the xls file.

Support for Excel Output Command files begins with version 7.52 revision E. No support is provided in prior versions/revisions.

Accounting reports are not supported.

This document is located in K:\WORD\HELP DOCS\ExcelOutputCommandFile.doc

What is an EFX file?

An EFX file is constructed similar to EF files in that they have ~IN~ and ~OUT~ code segments which will be executed before and after the actual Excel generation, respectively. It does, however, have the following exceptions:

- The EFX file does not require a corresponding FRX file. Instead, the EFX file itself will appear as a custom report on the report criteria dialog and will be executed when it is displayed/printed. It still needs a custom name, as listed for the FRX files above. When running an EFX report, select Display (not Excel).

- The ~IN~ code segment executes before the Excel object is created and allows definition of variables that defines various aspects of the excel files such as Report Titles, row spacing, etc. It can also be used to provide table linking or filtering if desired.

- Version 8.22A and higher support a ~BEGIN~ code segment that executes immediately after an Excel worksheet has been created but before any data has been inserted. This can be used to execute specialized code that directly manipulates the worksheet in an object oriented manner.

- The EFX file MUST have a segment labeled ~FIELDS~ which contains a list of fields or expressions which represent columns in the generated Excel file. The ~FIELDS~ segment is explained in greater detail later in this document.

- Version 8.22A and higher support a ~DONE~ code segment that executes after data has been inserted into the Excel worksheet but immediately before the worksheet is saved to file. Like the ~BEGIN~ segment this can be used to execute specialized code that directly manipulates the worksheet in an object oriented manner.

- The ~OUT~ code segment executes after the Excel object is destroyed and allows for clean-up of any changes that may have been made to the data environment the ~IN~ segment.

Variables exposed during the ~IN~ process

The Excel generation process exposes several variables during the ~IN~ process to allow alteration of overall behavior and to define titles to appear at the top of the spreadsheet.

xlsRowSpacing = 1	: Adds a blank row between data records.	Default: 0	Max: 10
xlsMainTitle = "Report Title"	: Topmost title. Is bold with a larger font size.		
xlsSubTitle1 = "Sub-Title Line 1"	: Is bold with regular size font.		
xlsSubTitle2 = "Sub-Title Line 2"	: Is bold with regular size font.		
xlsSubTitle3 = "Sub-Title Line 3"	: Is bold with regular size font.		
xlsColHeadUnderline = .F.	: Determines whether column titles are underlined.	Default: .T.	
xlsGroupExpr = " "	: Expression which determines when subtotaling should be performed.		
xlsGroupRowSpacing=1	: Number of blank row between a subtotal and the next grouping.		
xlsMemoParse = .F.	: Triggers parsing of memo field lines into separate rows when .T.		
xlsMemoLength = 40	: Largest number of characters per memo line.		
xlsMemoDelimiter = " "	: Character used to delimit items in the memo field.		

Notes: ***xlsMemoLength*** *&* ***xlsMemoDelimiter*** *will have no effect unless* ***xlsMemoParse*** *is .T.*
xlsGroupExpr *&* ***xlsGroupRowSpacing*** *support begins with version 7.72K*

Variables exposed in the ~BEGIN~ and ~DONE~ processes: ***(Version 8.22A & higher only)***

The Excel generation process exposes several variables during the ~BEGIN~ and ~DONE~ processes to allow direct manipulation of the Excel objects. Familiarity with the Excel programmatic object model is imperative for successful object usage.

oExcel	: Object reference to the Excel application.
oBook	: Object reference to the workbook that was created within Excel to hold the data.
oAS	: Object reference to the active sheet of the workbook into which data will be/was inserted.

Defining fields & expressions in the ~FIELDS~ segment

Fields and/or expressions to be output to the Excel file are defined in the ~FIELDS~ segment. Multiple fields and/or expressions can even be defined to be exported within the same column. Typically only a single field would be output per column. Here is a simple ~FIELDS~ definition:

```
~FIELDS~
Partno
Descript
```

In this case only the part number and the description would be exported. Since the name of the table from which the data is being taken has not been specified it is assumed to be from the table open in the current work area. However, if a relation between two table is established within the ~IN~ process you may need to identify the tables explicitly.

```
~FIELDS~
Sales.Partno
Partmast.Descript
```

Expressions can be defined and the result will be exported. Expressions MUST be preceded with EXPR:

```
~FIELDS~
Sales.Partno
Partmast.Descript
EXPR: Sales.OrQtyReq * Partmast.Cost
```

Multiple fields and expressions can be defined for the same column. This allows similar data to be "stacked" vertically rather than stretched out horizontally. A **semicolon** must separate the fields or expressions.

```
~FIELDS~
Sales.Partno
Partmast.Descript
EXPR: Sales.OrQtyReq * Partmast.Cost
PartMast.Manufacter;PartMast.MFG2;PartMast.MFG3
```

Fields and expressions can be stacked simultaneously.

```
~FIELDS~
Sales.Partno
Partmast.Descript
EXPR: Sales.OrQtyReq * Partmast.Cost
PartMast.Manufacter;PartMast.MFG2;PartMast.MFG3
PartMast.Vendor1;EXPR: Addrbook.Phone+"  x"+Addrbook.Ext
```

Important Note: If invalid fields or expressions are defined in ~FIELDS~ then the user will be notified and provided with the opportunity to display/print an error report.   The process can then be cancelled or the errors can be ignored and generation of the Excel file will continue without the offending definitions.

Function usage within Expressions

Generally any valid VFP or SAI function can be used within an expression so long as it makes contextual sense. Since each line within the ~FIELDS~ block represents a single excel column definition, complicated multi-line expressions are not permitted.

The EFX parser also exposes a dependant function, EO_PointerMoved(), used to identify movement of record pointers within one-to-many related tables.   The function accepts a character string as a parameter that represents the name of the table whose record movement status is questioned.   If a parameter is NOT passed then the primary table is assumed.   See example # 5 below as a sample of usage.

Field Control Parameters

Field control parameters can be added to the field definitions to "tweak" the look of the exported data.

/BackColor {number} Background color for the column's cells.

/Bold Triggers the font to display as bold.

/Caption {text} Caption for the column

/Color {number} Color value of the column's font.

Some of the common color codes are:

Black (Default)	0	Dark Red........	128	Dark Blue......	8388608
Light Grey......	12632256	Light Green.....	49152	Gold...........	32896
Dark Grey......	8421504	Dark Green......	32768	Purple.........	16711808
Light Red.......	192	Light Blue.......	12582912	Brown.........	16512

/Font {Font Name} The font to be used for the column.

/Format {Format Codes} Allows an Excel NumericFormat format mask to be specified for the column. (do not include the ")

Code	Description	Example value	Example string	Example output
"General"	Resets to the default format.	1234.5	"General"	1234.5
#	Displays a number (blank if a leading or trailing 0).	1234.5	"#####.#"	1234.5
0	Displays a number, including leading or trailing 0's.	1234.5	"00000.00"	01234.50
# 0	Combination of the above.	1.23	"###0.0000"	1.2300
,	Adds a Thousands separator.	1234.5	"##,###.##"	1,234.5
%	Displays numbers as a percentage.	.08	"##%"	8%
$	Inserts the dollar sign.	1.25	"$##.00"	$1.25
€	Inserts the Euro symbol.	3.00	CHR(128) + "##.00"	€3.00
M	Displays the month as a number from 1–12.	10/22/99	"M"	10
Mmm	Displays the month as a three-character abbreviation.	10/22/99	"Mmm"	Oct
D	Displays the day as a number from 1–31.	10/22/99	"D"	22
Ddd	Displays the day as a three-character day of week.	10/22/99	"Ddd"	Fri
Yy	Displays a two-digit year.	10/22/99	"Yy"	99

There are many more available; see the Help topic "About number formats" in the regular Excel Help file.

/HAlign {Alignment} Allows a horizontal alignment to be specified for the cells of the column.

Left - Left Aligned
Right - Right Aligned
Center - Centered
Auto - [default] Excel selects the best alignment based on the data

/VAlign {Alignment} Allows a vertical alignment to be specified for the cells of the column.

Top - [default] Top Aligned
Center - Centered
Bottom - Bottom Aligned

/Size {number}	The size of the font for the column.
/SubTotal	The column will be summed when the Group Expression changes. [Requires a valid group expression defined via **xlsGroupExpr**]
/Total	The column will be summed at the bottom.
/Trim	Trims leading & trailing spaces from Text or Memo data.
/Width {number}	Sets the column width to a specific value.
/Wrap	Turns on word wrapping for the column. It should only be used if the width has also been set via the /Width property.

The result of a function or expression can be used as a parameter setting. Here are some examples:

/Bold Expr: File('boldxls.prg') -- Uses the presence of a file to determine if the column is bold.

/Color Expr: RGB(128,0,0) -- Defines the color via separate Red, Green, Blue values.

/Width Expr: CalculateWidth(FieldName) -- Calls a function (which doesn't really exist) which could (if it DID exist) calculate the column width based on the field characteristics.

Example of an EFX file:

This sample inventory Excel output command file (CUSPAR01.EFX) shows inventory information and related vendor address information from the Address Book. (Note: No need to SET RELATION if the second table is a child table, ex. Partmast.Manufacter)

(**Important Note**: This example was developed for versions 8.20 and earlier. It WILL fail if it is used in version 8.22A or higher.)

```
~IN~

USE ADDRBOOK INDEX BYIDNO IN 0
SET RELATION TO UPPER(ID1) INTO ADDRBOOK
xlsMainTitle = "Inventory Vendor List"
xlsSubTitle1 = "Date Generated: "+DTOC(DATE())

~FIELDS~

Partno /Caption Part Number /Size 12 /Color 8388608
Descript /Caption Description
EXPR: onhand+Area2qty+Area3qty+Area4qty+Area5qty+Area6qty /Caption Inventory Qty
Manufacter;Mfg2;Mfg3;Mfg4;Mfg5;Mfg6;Mfg7;Mfg8;Mfg9 /Caption Manufacturers
Addrbook.Name /Caption Vendor
EXPR: Addrbook.Phone+"  x"+Addrbook.Extension /Caption Vendor's Phone

~OUT~

USE IN SELECT("ADDRBOOK")
```

Example 2 of an EFX file:

This sample BOM Excel output command file (CUSBOM01.EFX) shows only specific information from the temporary BOM report table.

(**Important Note**: This example was developed for versions 8.20 and earlier. It WILL fail if it is used in version 8.22A or higher.)

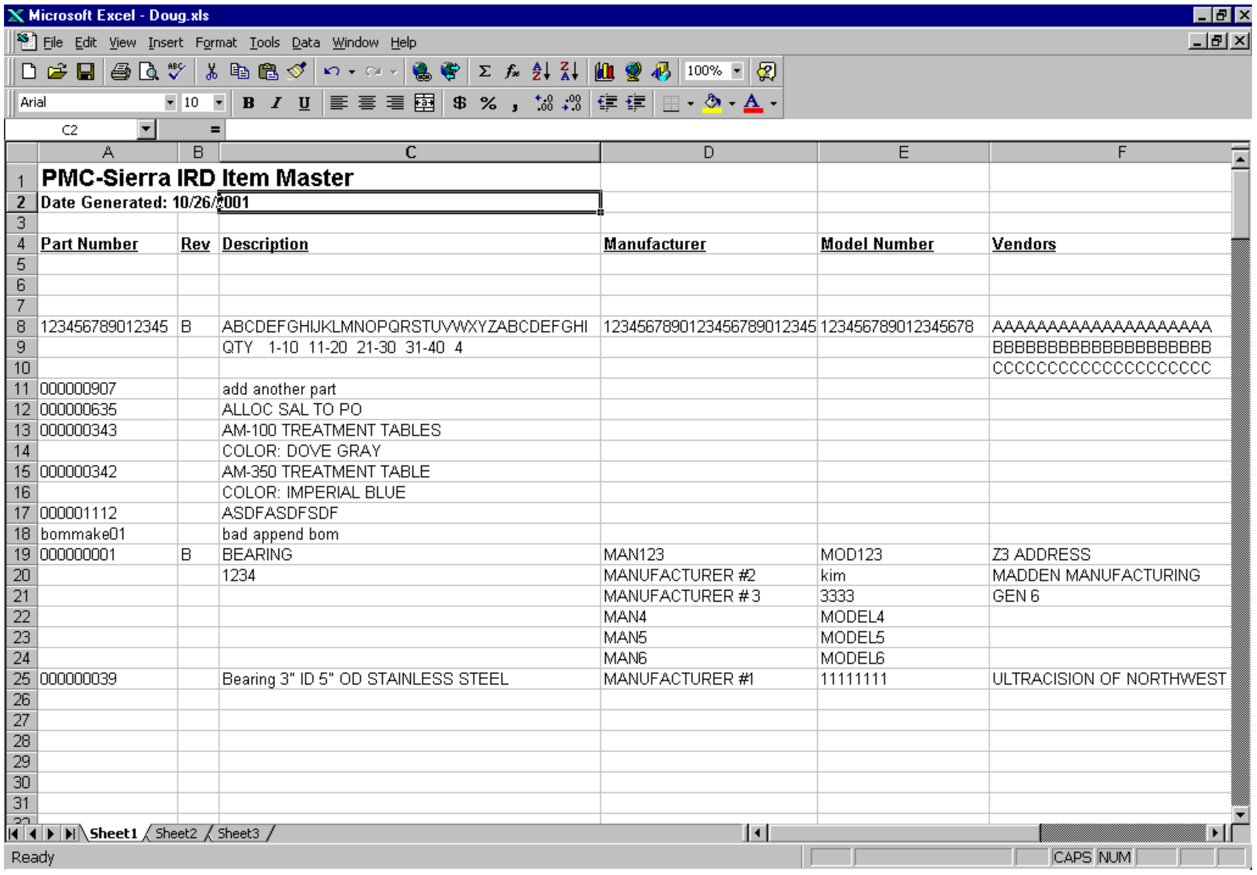
```
~IN~

xlsMainTitle = "Fictitious Company IRD Item Master"
xlsSubTitle1 = "Date Generated: "+DTOC(DATE())
xlsSubTitle2 = "BOM #:  "+MBOMNO
xlsSubTitle3 = "BOM Description: "+MBOMNAME
RELATE("PartMast")


~FIELDS~

PARTNO /Caption Part Number
PartMast.REVLEVEL /Caption Rev
PartMast.DESCRIPT;EXPR: LEFT(PartMast.ALTPARTNO,34) /Caption Description
MANUFACT; MANUFACT2; MANUFACT3; MANUFACT4; MANUFACT5; MANUFACT6 /Caption Manufacturer
MODELNO;MODELNO2;MODELNO3;MODELNO4;MODELNO5;MODELNO6 /Caption Model Number
PartMast.VENDOR1; PartMast.VENDOR2; PartMast.VENDOR3 /Caption Vendors
PartMast.STDCOST /Caption Standard Cost
```

NOTE: To test any of these examples, copy and paste the code into a text file with the indicated filename.



Microsoft Excel - Doug.xls
File Edit View Insert Format Tools Data Window Help
100%
Arial
10
C2
=
Sheet1
Sheet2
Sheet3
Ready
CAPS NUM

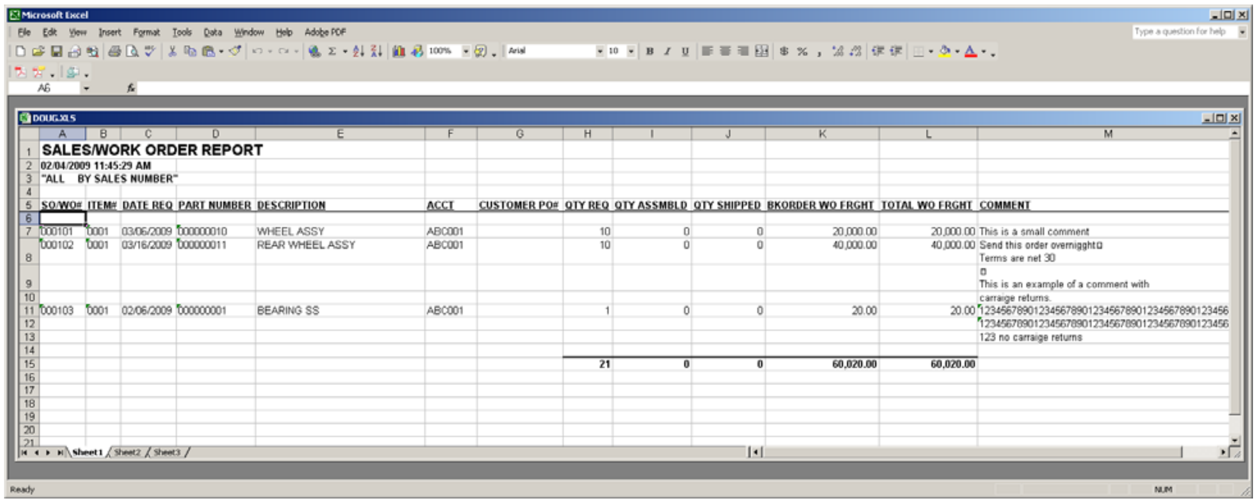
	A	B	C	D	E	F
1	**PMC-Sierra IRD Item Master**					
2	**Date Generated: 10/26/2001**					
3						
4	**Part Number**	**Rev**	**Description**	**Manufacturer**	**Model Number**	**Vendors**
5						
6						
7						
8	123456789012345	B	ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHI	1234567890123456789012345	123456789012345678	AAAAAAAAAAAAAAAAAAAA
9			QTY 1-10 11-20 21-30 31-40 4			BBBBBBBBBBBBBBBBBBBB
10						CCCCCCCCCCCCCCCCCCCC
11	000000907		add another part			
12	000000635		ALLOC SAL TO PO			
13	000000343		AM-100 TREATMENT TABLES			
14			COLOR: DOVE GRAY			
15	000000342		AM-350 TREATMENT TABLE			
16			COLOR: IMPERIAL BLUE			
17	000001112		ASDFASDFSDF			
18	bommake01		bad append bom			
19	000000001	B	BEARING	MAN123	MOD123	Z3 ADDRESS
20			1234	MANUFACTURER #2	kim	MADDEN MANUFACTURING
21				MANUFACTURER # 3	3333	GEN 6
22				MAN4	MODEL4	
23				MAN5	MODEL5	
24				MAN6	MODEL6	
25	000000039		Bearing 3" ID 5" OD STAINLESS STEEL	MANUFACTURER #1	11111111	ULTRACISION OF NORTHWEST
26						
27						
28						
29						
30						
31						

Example 3 of an EFX file:

This sample adds the comments from sale.fpt memo file and also includes totals at the bottom of the xls report. To display the comment in one long row, use EXPR: CHRTRAN(Comment,CHR(10), " ") .

```
~IN~
xlsMainTitle = "SALES/WORK ORDER REPORT"
xlsSubTitle1 = TTOC(DATETIME())
xlsSubTitle2 = M
xlsMemoParse = .T.
xlsMemoLength = 46

~FIELDS~
Sono /Caption SO/WO#
ItemNo /Caption ITEM#
DateReq /Caption DATE REQ
Partno /Caption PART NUMBER
Descript /Caption DESCRIPTION
Accountno /Caption ACCT
CustPoNo /Caption CUSTOMER PO#
OrQtyReq /Caption QTY REQ /Total
QtyAssm /Caption QTY ASSMBLD /Total
QtyShip /Caption QTY SHIPPED /Total
EXPR: ROUND(CalcItem2(SalePrice, OrQtyReq-QtyShip, TaxR, 0, Discount, TaxFreight, TaxDiscnt), 2) /Caption BKORDER WO FRGHT /Format ##,###,###.00 /Total
EXPR: ROUND(CalcItem2(SalePrice, OrQtyReq, TaxR, 0, Discount, TaxFreight, TaxDiscnt), 2) /Caption TOTAL WO FRGHT /Format ##,###,###.00 /Total
Comment /Caption COMMENT
```



SALES/WORK ORDER REPORT
02/04/2009 11:45:29 AM
"ALL BY SALES NUMBER"

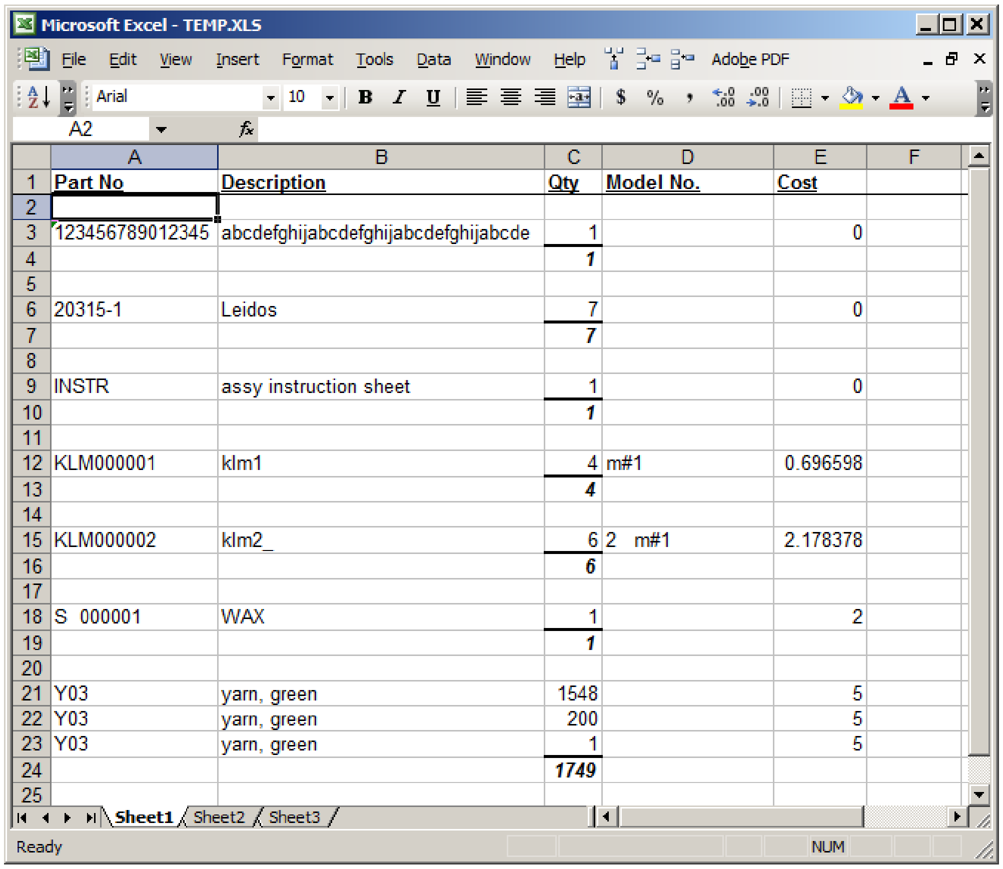
SO/WO#	ITEM#	DATE REQ	PART NUMBER	DESCRIPTION	ACCT	CUSTOMER PO#	QTY REQ	QTY ASSMBLD	QTY SHIPPED	BKORDER WO FRGHT	TOTAL WO FRGHT	COMMENT
000101	0001	03/06/2009	000000010	WHEEL ASSY	ABC001		10	0	0	20,000.00	20,000.00	This is a small comment
000102	0001	03/16/2009	000000011	REAR WHEEL ASSY	ABC001		10	0	0	40,000.00	40,000.00	Send this order overnigght□ Terms are net 30
												□ This is an example of a comment with
												carraige returns.
000103	0001	02/06/2009	000000001	BEARING SS	ABC001		1	0	0	20.00	20.00	1234567890123456789012345678901234567890123456
												1234567890123456789012345678901234567890123456
												123 no carraige returns
							21	**0**	**0**	**60,020.00**	**60,020.00**	

Example 4 of an EFX file:

This sample sends the MRP Buy report to Excel, along with some Partmast data.
The Actqty is subtotaled by Partno.

```
~IN~
xlsGroupExpr = "partno"

~FIELDS~
partno /Caption Part No
actdescr /Caption Description
actqty /Caption Qty /Subtotal
partmast.modelno /Caption Model No.
partmast.cost /Caption Cost
```



Microsoft Excel - TEMP.XLS
File Edit View Insert Format Tools Data Window Help Adobe PDF
Arial 10
A2
Sheet1 Sheet2 Sheet3
Ready NUM

	A	B	C	D	E	F
1	Part No	Description	Qty	Model No.	Cost	
2						
3	123456789012345	abcdefghijabcdefghijabcdefghijabcde	1		0	
4			*1*			
5						
6	20315-1	Leidos	7		0	
7			*7*			
8						
9	INSTR	assy instruction sheet	1		0	
10			*1*			
11						
12	KLM000001	klm1	4	m#1	0.696598	
13			*4*			
14						
15	KLM000002	klm2_	6	2 m#1	2.178378	
16			*6*			
17						
18	S 000001	WAX	1		2	
19			*1*			
20						
21	Y03	yarn, green	1548		5	
22	Y03	yarn, green	200		5	
23	Y03	yarn, green	1		5	
24			*1749*			
25						

Example 5 of an EFX file:

This example is a pared down Inventory Movement report with movement details. There is a one-to-many relationship from the primary table (Movement) into a child table (MoveDetail) with skip set to the child table. The child table is then related into its own child table (StockTra).

The EO_PointerMoved() function is used in this example to detect when movement within the primary table occurs so that duplicate parent data is NOT included in the output.

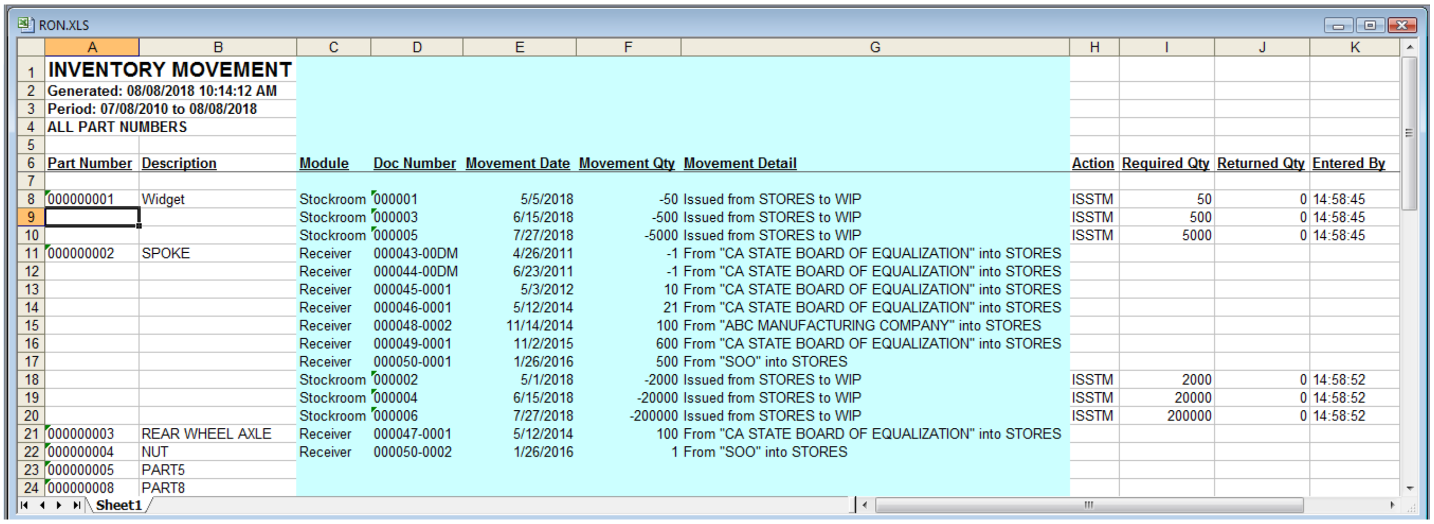
```
~IN~
IF USED("MoveDetail")
        LOCAl lnOldArea
        lnOldArea = SELECT()
        USE StockTra IN 0 ORDER StockNdx
        SELECT MoveDetail
        SET RELATION to UPPER(ALLTRIM(MoveDetail.Identifier)) INTO StockTra
        SELECT (lnOldArea)
        SET SKIP TO MoveDetail
ENDIF

~FIELDS~
EXPR: IIF(EO_PointerMoved(), PartNo, "") /Caption Part Number
EXPR: IIF(EO_PointerMoved(), Descript, "") /Caption Description
*EXPR: IIF(EO_PointerMoved(), InitQty, .NULL.) /Caption Initial Qty

EXPR: IIF(USED("MoveDetail") AND !EOF("MoveDetail"), MoveDetail.Module, "") /Caption Module /BackColor 16777164
EXPR: IIF(USED("MoveDetail") AND !EOF("MoveDetail"), MoveDetail.Identifier, "") /Caption Doc Number /BackColor 16777164
EXPR: IIF(USED("MoveDetail") AND !EOF("MoveDetail"), MoveDetail.MoveDate, .NULL.) /Caption Movement Date /BackColor 16777164
EXPR: IIF(USED("MoveDetail") AND !EOF("MoveDetail"), MoveDetail.Quantity, "") /Caption Movement Qty /BackColor 16777164
EXPR: IIF(USED("MoveDetail") AND !EOF("MoveDetail"), MoveDetail.Detail, "") /Caption Movement Detail /BackColor 16777164

EXPR: IIF(USED("MoveDetail") AND MoveDetail.Module="Stockroom", StockTra.Action, "") /Caption Action
EXPR: IIF(USED("MoveDetail") AND MoveDetail.Module="Stockroom", StockTra.QtyReq, .NULL.) /Caption Required Qty
EXPR: IIF(USED("MoveDetail") AND MoveDetail.Module="Stockroom", StockTra.ReturnQty, .NULL.) /Caption Returned Qty
EXPR: IIF(USED("MoveDetail") AND MoveDetail.Module="Stockroom", StockTra.EnterBy, "") /Caption Entered By

~OUT~
USE IN SELECT("STOCKTRA")
```



RON.XLS

	A	B	C	D	E	F	G	H	I	J	K
1	**INVENTORY MOVEMENT**										
2	**Generated: 08/08/2018 10:14:12 AM**										
3	**Period: 07/08/2010 to 08/08/2018**										
4	**ALL PART NUMBERS**										
5											
6	**Part Number**	**Description**	**Module**	**Doc Number**	**Movement Date**	**Movement Qty**	**Movement Detail**	**Action**	**Required Qty**	**Returned Qty**	**Entered By**
7											
8	000000001	Widget	Stockroom	000001	5/5/2018	-50	Issued from STORES to WIP	ISSTM	50	0	14:58:45
9			Stockroom	000003	6/15/2018	-500	Issued from STORES to WIP	ISSTM	500	0	14:58:45
10			Stockroom	000005	7/27/2018	-5000	Issued from STORES to WIP	ISSTM	5000	0	14:58:45
11	000000002	SPOKE	Receiver	000043-00DM	4/26/2011	-1	From "CA STATE BOARD OF EQUALIZATION" into STORES				
12			Receiver	000044-00DM	6/23/2011	-1	From "CA STATE BOARD OF EQUALIZATION" into STORES				
13			Receiver	000045-0001	5/3/2012	10	From "CA STATE BOARD OF EQUALIZATION" into STORES				
14			Receiver	000046-0001	5/12/2014	21	From "CA STATE BOARD OF EQUALIZATION" into STORES				
15			Receiver	000048-0002	11/14/2014	100	From "ABC MANUFACTURING COMPANY" into STORES				
16			Receiver	000049-0001	11/2/2015	600	From "CA STATE BOARD OF EQUALIZATION" into STORES				
17			Receiver	000050-0001	1/26/2016	500	From "SOO" into STORES				
18			Stockroom	000002	5/1/2018	-2000	Issued from STORES to WIP	ISSTM	2000	0	14:58:52
19			Stockroom	000004	6/15/2018	-20000	Issued from STORES to WIP	ISSTM	20000	0	14:58:52
20			Stockroom	000006	7/27/2018	-200000	Issued from STORES to WIP	ISSTM	200000	0	14:58:52
21	000000003	REAR WHEEL AXLE	Receiver	000047-0001	5/12/2014	100	From "CA STATE BOARD OF EQUALIZATION" into STORES				
22	000000004	NUT	Receiver	000050-0002	1/26/2016	1	From "SOO" into STORES				
23	000000005	PART5									
24	000000008	PART8									

Sheet1