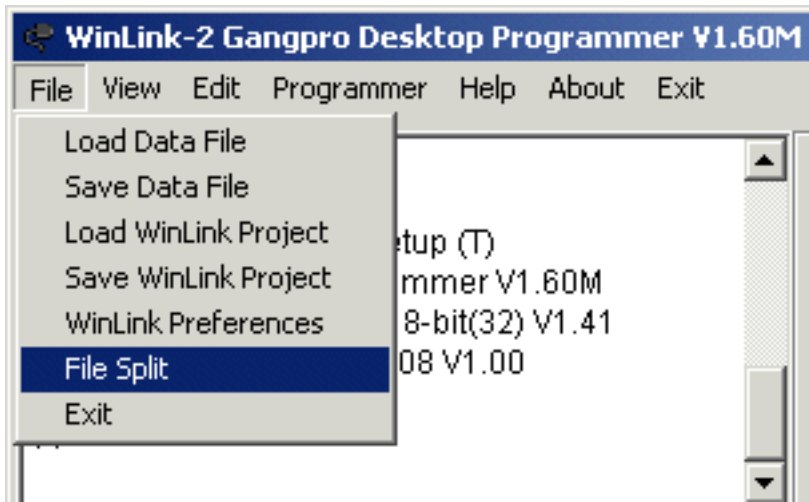


LOGICAL Help

File Split to Multiple Devices

The FILE SPLIT feature allows the EDITOR data to be interpreted as 16 or 32 bit data and split into multiple 8-bit memory devices. The file to be split is only loaded into the EDITOR once. The desired 'split' is selected 'on the fly' and programmed into all devices inserted in the platter.

Setting Up File Split Mode

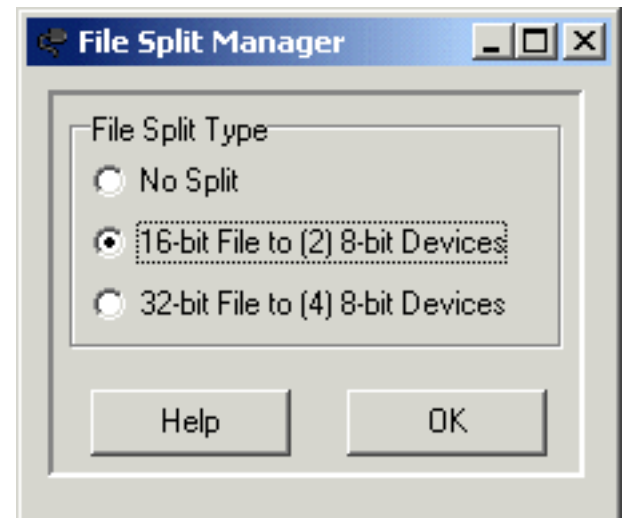


You can access the File Split Manager by clicking MAIN MENU | FILE | File Split .

This opens the File Split Manager as shown.

Click the desired File Split Option.

Click OK.





When the 16-bit or 32-bit File Split is selected, an additional panel is added below the Programmer Message Window.

The panel contains the radio buttons used to select the device from the

split.

For a 16-bit split (as shown) there will be two DEV buttons and for a 32-bit split there will be four DEV buttons.

Clicking a DEV button 'filters' or extracts the EDITOR data that is sent to the programmer.

This applies to all operations where data moves between the PROGRAMMER and the EDITOR.

(i.e. PROGRAM, VERIFY, and READ)

For a 16-bit split...

DEV#0 extracts only the data at EVEN EDITOR addresses.

DEV#1 extracts only the data at ODD EDITOR addresses.

For a 32-bit split...

DEV#0 extracts only the data where EDITOR address AND'ed with 0x03 == 0 .

DEV#1 extracts only the data where EDITOR address AND'ed with 0x03 == 1 .

DEV#2 extracts only the data where EDITOR address AND'ed with 0x03 == 2 .

DEV#3 extracts only the data where EDITOR address AND'ed with 0x03 == 3 .

16-bit example...

EDITOR 000000 = 0011223344556677

DEV#0 0000 = 00224466

DEV#1 0000 = 11335577

32-bit example...

EDITOR 000000 = 00112233445566778899AABBCCDDEEFF

DEV#0 0000 = 004488CC

DEV#1 0000 = 115599DD

DEV#2 0000 = 2266AAEE

DEV#3 0000 = 3377BBFF

First Time Step-by-Step Example

- 1) **Select Device** [...howto](#)
- 2) **Setup File Split**^{1,2} [...howto](#)
- 3) **Load Data File**³ [...howto](#)
- 4) **Select/Check the desired Split** [...howto](#)
- 5) **Insert blank devices**
- 6) **Program Devices with Current Split Data** [...howto](#)
- 7) **Remove Devices**
- 8a) **Program More Devices with Same Split...repeat steps 5-7**
- 8b) **Program Devices with Next Split...repeat steps 4-7**

Notes :

¹ You can setup or disable File Split at any time but doing it prior to loading data file makes address setup easier.

² Each time you exit the File Split Manager, DEV#0 is automatically selected .

³ Make sure the Up To File Adress value is large enough for the file. Clicking the RESET button automatically sets this value properly.

File Split Effects in Other Areas

Editor Checksum

When File Split is active, the EDITOR checksum calculation reflects only the currently selected split. The Device CHKSUM button still calculates the checksum of the data in the device.

Note : If you add the EDITOR checkums generated by selecting each split, this sum should equal the original FILE's checksum.

Data File Load

When File Split is active, the calculation performed by the RESET button is modified so that the FILE FILTER end address accomodates the increased file size required by multiple devices.

When File Split is active, the chksums for each split as well as the FILE chksum is displayed in the message window after the file loads.

Editor End Address

The EDITOR size is increased to accomodate larger data size required by multiple devices when File Split is active.

Front Panel

The Split Select Panel appears below the programmer message window when File Split is active.

Byte Ordering

All File Types are by nature 8-bit and have no intrinsic byte ordering for words larger than 8-bits. However the compiler/assembler/linker that generated the file implicitly assumes a byte ordering scheme. The target application hardware also assumes an ordering scheme. There is no way to determine (from the file) which ordering scheme is assumed in the FILE. It is up to YOU to make sure the proper data gets into the proper device.

Notice that by default the byte ordering is LIL-ENDIAN (the LS byte is located at the lower EDITOR address.

LIL-ENDIAN

16-bit

EDITOR 0000 = bits[7..0] -> DEV#0

EDITOR 0001 = bits[15..8] -> DEV#1

32-bit

EDITOR 0000 = bits[7..0] -> DEV#0

EDITOR 0001 = bits[15..8] -> DEV#1

EDITOR 0002 = bits[23..16] -> DEV#2

EDITOR 0003 = bits[31..24] -> DEV#3

This is typical for Intel based architectures.

BIG-ENDIAN

16-bit

EDITOR 0000 = bits[15..8] -> DEV#0

EDITOR 0001 = bits[7..0] -> DEV#1

32-bit

EDITOR 0000 = bits[31..24] -> DEV#0

EDITOR 0001 = bits[23..16] -> DEV#1

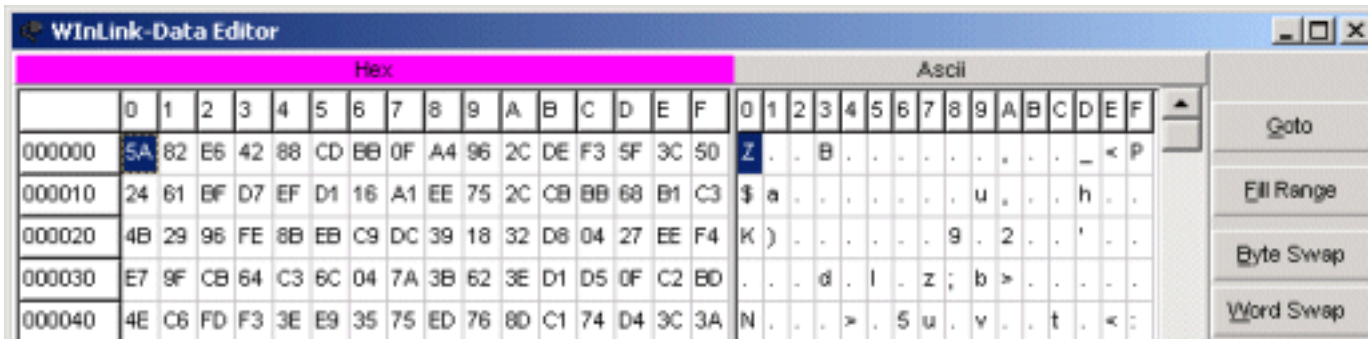
EDITOR 0002 = bits[15..8] -> DEV#2

EDITOR 0003 = bits[7..0] -> DEV#3

This is typical for Motorola based architectures.

Should it be necessary to interpret the EDITOR data as BIG-ENDIAN (the MS byte is located at the lower EDITOR address, use the BYTE SWAP and WORD SWAP(if 32-bit) buttons in the EDITOR to massage the byte order after loading the file. (*You could also simply physically re-arrange the devices in the target app*)

CONVERT BYTE-ORDER in the EDITOR



Use **BYTE SWAP** to convert 16-bit LIL-ENDIAN <-> BIG-ENDIAN

example...

EDITOR 000000 = 00112233 after **BYTE SWAP** EDITOR 000000 = 11003322

Use **BYTE SWAP** and **WORD SWAP** to convert 32-bit LIL-ENDIAN <-> BIG-ENDIAN

example...

EDITOR 000000 = 00112233 after **BYTE SWAP** EDITOR 000000 = 11003322

EDITOR 000000 = 11003322 after **WORD SWAP** EDITOR 000000 = 33221100

Tips & Tricks

Programming devices from a file (via the Winlink2 EDITOR) is considerably slower than copying devices from a MASTER. If your programming task requires large quantities of split file devices, it is recommended that you produce a single set of MASTER devices (one for each split) and then use the COPY operation to produce the desired quantity of each split (from the appropriate MASTER).