```
                  Identification
                  --------------

Product Code:     MFE-8E-XMOD-B

Product Name:     OS/8 XMODEM Device Handlers

date created:     September 4, 2020

Maintainer:       Martin Eberhard

Author:           Martin Eberhard
```

1. ABSTRACT

XMR and XMT are OS/8 device handler groups that send and receive data via a KL8-E type serial port, using the Xmodem protocol. XMR receives a file into OS/8, and XMT0 transmits a file out of OS/8. XMTA also transmits a file out of OS/8, but strips the most-significant bit of every 8-bit byte. (XMTA is intended for ASCII files.)

The purpose of these handlers is to allow files to be transferred from any other machine to the OS/8 environment, and from the OS/8 environment to any other machine. This is useful for backing up and restoring files, for loading into the OS/8 environment archived programs and programs that were developed on another machine (perhaps with a cross-assembler such as PALBART), and for transferring OS/8 text files to be printed on another machine.

The Xmodem protocol is especially useful because it provides in-band flow control, in additional to its error checking and error-retransmit capabilities. (Flow control allows the transfer of files larger than the receiving end's buffer.) Suitable Xmodem file transfer programs (such as Teraterm) are widely available for many current operating system environments (including Windows, MAC OS, and Linux) eliminating the need for custom software on the remote machine.

2. REQUIREMENTS

2.1 EQUIPMENT

* A pdp-8 (any variant)
* A serial device (KL8-E or equivalent)
* A mass storage device (disk or magnetic tape).

Timing values for timeouts in this document assume a PDP8-E. However, all timing values will work for other PDP-8 variants.

2.2 DEVICE HANDLER SPECIFICS

This table summarizes the three Xmodem device handlers.

Handler	Description	Group	File	Type	Entry Point Offset	Pages
XMR	Xmodem Receive	XMR	XMR.BN	56	070	2
XMT0	8-bit Xmodem Transmit	XMT	XMT.BN	55	057	2
XMTA	ASCII Xmodem Transmit	XMT	XMT.BN	55	062	

2.2.1 Device Handler versions

The version numbers of XMR, XMT0, and XMTA are stored in their entry point addresses. This document refers specifically to version 0100 of XMR, and version 0101 of the XMT handlers, though it is hopefully relevant for later versions too.

2.2.1 Device Handler sizes

XMR and the XMT handlers are 2-page OS/8 device handlers, requiring 512(10) words of memory each when used. The XMT handlers (XMT0 and XMTA) are co-resident with each other.

2.2.2 Device Handler names

The file names for these handlers (as needed for loading into Build) are XMR.BN and XMT.BN.

XMR's group name and device name are both XMR. The XMT handlers' group name is XMT, with device names XMT0 and XMTA.

2.2.3 Device Handler Types

The device type of XMR is 56. The device type of both XMT0 and XMTA is 55. (See DEC-S8-OSHBA-A-D page 2-59) None identifies itself as a file-structured handler. XMR is a read-only handler and the XMT handlers are both write-only.

2.2.4 Device Entry Point Offsets

The entry point offsets for all of these handlers were chosen so as not to conflict with any standard OS/8 file-structured device (though this is not strictly necessary). (See DEC-S8-OSHBA-A-D page 2-60)

The entry point offset for XMR version 0100 is 0070. The entry point offset for XMT0 version 0101 is 0057. The entry point offset for XMTA version 0101 is 0062.

2.2.5 IOT Address

As released, these handlers communicate via a KL8-E type device at IOT base address 0300. This is easily changed with one line in the source code of each handler, followed by reassembly.

3. LOADING PROCEDURE

This section explains how to load these handlers from paper tapes of the binaries, XMR.BN and XMT.BN. In these examples, the underlined text is to be typed by the user. {Text in curly braces indicates a user action other than typing, or a comment.}

3.1 LOADING THE HANDLER BINARY IMAGES ONTO OS/8'S SYSTEM DEVICE

Load XMR.BN onto the system device via a paper tape reader using PIP:

```
.R PIP
        {Load XMR.BN tape in the reader}
*XMR.BN<PTR:/B
        {Start the reader}
        {Stop the reader when the file is loaded}

        {Load XMT.BN tape in the reader}
```

```
   *XMT.BN<PTR:/B
           {Start the reader}
           {Stop the reader when the file is loaded}
   *^C
           {Actually control-C}
   .
```

3.2 INSTALLING XMR AND XMT DEVICE HANDLERS INTO OS/8

Use “BUILD” to install these device handlers, as you would any other device handler:

```
   .RUN SYS BUILD
   $LOAD XMR.BN
   $LOAD XMT.BN
   $INSERT XMR
   $INSERT XMT0
   $INSERT XMTA
   $BOOTSTRAP
   SYS BUILT
   .SAVE SYS BUILD
   .
```

Note: if this is your first time using BUILD on this DECtape, then the configuration that Build has may not be the one that your OS/8 is using. Use BUILD’s “PRINT” command to examine its configuration, and make the necessary corrections (especially for the SYS: device) before using the BOOTSTRAP command.

4. OPERATION

These handlers operate as normal non-file-structured device handlers, capable of transferring files using PIP or other OS/8 programs.

4.1 LOADING FILES FROM A REMOTE MACHINE ONTO OS/8 WITH XMR

```
   .R PIP
       {Start Xmodem file-send on the sending machine. For example,
        using TeraTerm on Windows, choose File>Transfer>Xmodem>send.}
   *DEV:<MYFILE.NN<XMR/I
       {Generally use /I, regardless of file type. You can use /B for
        binary files, which will cause PIP to adjust the length of the
        leaders in the file to the OS/8 “standard” lengths.}
   *^C
       {Actually Control-C}
```

Note that XMR will append a ^Z character (232 octal) to the end of the file, and pad out the OS/8 buffer with nulls. (See DEC-S8-OSSMB-A-D page 5-3 item 11.)

4.2 SENDING OS/8 FILES WITH XMT0

To save any type of file (ASCII, binary, image, or anything else) on a remote machine:

```
.R PIP
*XMT0<DEV:MYFILE.NN/I
    {Always use /I, regardless of file type}
    {Wait for OS/8 to start fetching the file.}
    {Start Xmodem receive on the receiving machine. For example,
     using TeraTerm on Windows, choose File>Transfer>Xmodem>Receive.}
*^C {Actually control-C}
```

4.2.1 Sending ASCII Files

OS/8 ASCII files usually have the most-significant bit (the parity bit) set for all characters. This is incompatible with most Windows and Mac editors that expect this bit to be cleared for the normal ASCII characters. (The high bit is commonly used to access an extended character set.)

XMTA clears the parity bit on each byte it transmits, creating a file that can be used on machines that expect the parity bit to be 0 for normal ASCII characters. XMTA works the same as XMT0.

```
.R PIP
*XMTA<DEV:MYFILE.NN/I
    {Always use /I, regardless of file type}
    {Wait for OS/8 to start fetching the file.}
    {Start Xmodem receive on the receiving machine. For example,
     using TeraTerm on Windows, choose File>Transfer>Xmodem>Receive.}
*^C
    {Actually control-C}
.
```

If you used XMT0 to send the file, you can strip the parity bits off on the remote machine, using for example Microsoft Word:

1. Open the received file with Word
2. Select “Other Encoding”>”US-ASCII”
3. Save the file as a plain text (.txt) file

4.3 ABORTING A TRANSFER

You can abort a transfer that is using any of these handlers by typing ^C (control-C) on the OS/8 console. This will abort to the OS/8 system monitor (via a jump to 07600). Note that this will not work if you have reassigned the OS/8 console to anything other than the standard device at IOT 0030. (See DEC-S8-OSSMB-A-D page 5-2 item 4.)

5. CALLING DEVICE HANDLERS

See DEC-S8-OSSMB-A-D, chapters 4 and 5.

5.1 DEVICE HANDLER INITIALIZATION

All of these handlers require the first call to the handler to have the OS/8 record number to be equal to 0. This tells the handler to perform initialization. (See DEC-S8-OSSMB-A-D page 5-1 item 2.)

5.2 CLOSING DEVICE HANDLERS

Although it is good practice always to close a device handler at the end of a session, only the XMT handlers actually require being closed. (Closing an XMT handler tells the handler to send an Xmodem EOT, to terminate the Xmodem session.)

Close a handler by calling it with ARG(1) bits <1:5> and <9:11> equal to 0. (These handlers don’t actually look at bits <9:11>) (See DEC-S8-OSSMB-A-D page 4-3 item 5.)

Since calling a handler with the OS/8 record-count field equal to 0 closes the handler session, the buffer length cannot be a complete field of 4096(10) words. The largest possible buffer is 3840 (10) or 7400(8) words, with ARG(1) bits <1:5> = 37(8). (See DEC-S8-OSSMB-A-D page 4-3 items 1 AND 5. Note that ARG(1) bits <1:5> = 00 implies a 4096(10)-word transfer only for file-structured devices -- which XMR and XMT are not.)

5.3 DEVICE HANDLER EXITS

Aside from an abort by the user (section 4.3), these handlers always exit via the Error Return or the Normal Return, assuming the standard OS/8 handler calling sequence. (See DEC-S8-OSSMB-A-D page 4-1.)

5.3.1 XMR Normal Exit

XMR performs a normal exit by setting AC=0 and returning via the “Normal Return” address.

XMR exits normally when it has filled the OS/8 buffer completely.

5.3.2 XMR End-of-File Exit

XMR indicates the end-of-file by setting AC=0 and returning via the “Error Return” address.

XMR indicates End-Of-File when it receives an Xmodem EOT character. It writes a Control-Z character into the buffer, and zeroing the rest of the buffer before returning. (See DEC-S8-OSSMB-A-D, page 5-3, item 11.)

5.3.3 XMR Error Exit

XMR indicates a fatal error by setting AC=-1 and returning via the "Error Return" address.

XMR will request retries for bad Xmodem record headers and for checksum errors in the received record data, as per the Xmodem spec. The number of requested retries is unbounded. (This is contrary to the Xmodem spec, which specifies aborting after 10 retries.)

The following conditions cause XMR to indicate a fatal error:

1. The device handler was called for sending
2. The device handler was called with an odd number of 128-word records specified for the buffer
3. The device handler received an error-free Xmodem record with the wrong record number. (Per the Xmodem spec, XMR will just ignore a repeat of the most recent record received.)
4. Timeout after 8 seconds, waiting for the beginning of an Xmodem record, including the initial record
5. Timeout after 2 seconds, waiting for an Xmodem record byte

5.3.4 XMT Handler Exits

5.3.5 XMT Handler Normal Exit

The XMT handlers perform a normal exit by setting AC=0 and returning via the "Normal Return" address.

The XMT handlers exit normally when THE OS/8 buffer has been emptied.

5.3.6 XMT Handler End-of-File Exit

The XMT handlers will never exit indicating end-of-file.

5.3.7 XMT Handler Error Exit

The XMT handlers indicate a fatal error by setting AC=-1 and returning via the "Error Return" address.

The XMT handlers will resend the most recent Xmodem record if a NAK is received for that record instead of an ACK (as per the Xmodem spec). These handlers will allow any number of resend requests, contrary to the Xmodem spec, which specifies aborting after 10 retries.

The following conditions cause an XMT handler to indicate a fatal error:

1. The device handler was called for receiving
2. The device driver was called with an odd number of 128-word records specified for the buffer
3. Timeout after 60 seconds, waiting for an Xmodem ACK or NAK

6. OS/8 BUFFER STRUCTURE FOR DEVICE HANDLERS

See DEC-S8-OSSMB-A-D pages A-4 through A-5.

6.1 BUFFER SIZE

As mentioned previously, these handlers require the calling program to specify the OS/8 buffer as an even number of 128-word records. This is not much of a hardship because all OS/8 files comprise an integer number of 256-word blocks, each made up of two 128-word records. (See DEC-S8-OSSMB-A-D, page 4-3, item 3.)

6.2 HOW XMR PUTS DATA INTO THE OS/8 BUFFER

XMR receives Xmodem data as 8-bit bytes, and packs them into OS/8's buffer in groups of three bytes in the OS/8-standard way. Each Group of three Xmodem bytes becomes two OS/8 buffer bytes:

	0 3	4 11
Word 1:	Byte 3 <0:3>	Byte 1 <0:7>
Word 2:	Byte 3 <4:7>	Byte 2 <0:7>

When XMR receives the Xmodem EOT character (indicating the end of the file), it inserts a Control-Z character (232 octal) at the end of the buffer and fills the rest of the buffer with 0's, before returning and indicating end-of-file. (See DEC-S8-OSSMB-A-D 5-3 item 11)

Note that the ^Z character may be byte 3 of the last three-byte sequence, and will then be split into the high-order four bits of the previous two buffer words.

6.3 HOW THE XMT HANDLERS UNPACK DATA FROM THE OS/8 BUFFER

The XMT handlers unpack OS/8 buffer data in the same way as described in section 6.1, producing three Xmodem bytes for every two OS/8 buffer words. Because Xmodem records are 128 (8-bit) bytes long and because these handlers require the OS/8 buffer to be an even number of 128-word records long, the transmission will always end with a complete Xmodem record -- and so there is no need ever to pad the last Xmodem record.