Template for Functional Specifications

Following is a template for Functional Specifications. It should be used in conjunction with the ''Guidelines for Functional Specifications'' document to create functional specifications for Quarterdeck software.

The table of contents and index are quite important and should follow general compositional practices.

Any appendixes are not always considered part of the actual requirements specification and are not always necessary. They may include

a) Sample I/O formats, descriptions of cost analysis studies, or results of user surveys
b) Supporting or background information that can help the readers of the FS
c) A description of the problems to be solved by the software
d) Special packaging instructions for the code and the media to meet security, export, initial loading, or other requirements

When appendixes are included, the FS should explicitly state whether or not the appendixes are to be considered part of the requirements.

Table of Contents

Template for Functional Specifications

Following is a template for Functional Specifications. It should be used in conjunction with the “Guidelines for Functional Specifications” document to create functional specifications for Quarterdeck software.

The table of contents and index are quite important and should follow general compositional practices.

Any appendixes are not always considered part of the actual requirements specification and are not always necessary. They may include

a) Sample I/O formats, descriptions of cost analysis studies, or results of user surveys b) Supporting or background information that can help the readers of the FS c) A description of the problems to be solved by the software d) Special packaging instructions for the code and the media to meet security, export, initial loading,

or other requirements

When appendixes are included, the FS should explicitly state whether or not the appendixes are to be considered part of the requirements.

Table of Contents 1. Introduction 2

1.1 Purpose 2 1.2 Definitions, acronyms, and abbreviations 2 1.3 References 2 2. Overall description 4 2.1 Product perspective 4 2.2 Product components 5 2.3 Product constraints 5 2.4 Proposed future requirements 8 3. Specific requirements 9

3.1 Specific requirement #1 pp 3.2 Specific requirement #2 pp

•

•

• 3.n Specific requirement #n pp Appendixes pp Index pp

1. Introduction

The introduction of the FS should provide an overview of the entire FS. It should contain the following subsections

a) Purpose
b) Definitions, acronyms, and abbreviations
c) References

1.1 Purpose

This subsection should

a) Describe briefly the purpose of the software
b) Identify the intended audience for the software

Example:

QEMM 7.53 provides the user with as much conventional memory as possible for running programs while also providing all of the published memory management services used by DOS programs and devices. The product also contains features to display the current DOS and Windows memory configurations to help the end-user identify problems and opportunities for improving the system.

QEMM 7.53 is intended for retail distribution into the DOS and MS Windows utility market. It is expected to be an attractive product to the power user, the computer game player, and the network and multimedia users who need extra conventional memory to run their DOS programs.

1.2 Definitions, acronyms, and abbreviations

This subsection should provide the definitions of all terms, acronyms, and abbreviations required to interpret properly the FS. This information may be provided by reference to one or more appendixes in the FS or by reference to other documents.

Example:

CPU: the computer chip that is the Central Processing Unit of the PC. For example, the Intel 80386SX, 80386DX, i486SX, i486DX, or Pentium processors and the compatible processors made by AMD, Cyrix, and others.

TSR: a "Terminate and Stay Resident" program, typically loaded in the AUTOEXEC.BAT file or other batch file executed during the boot sequence, which loads into memory, but then terminates back to the command processor prompt, leaving a portion of itself resident in memory to provide some service to the user. Examples of TSRs are network drivers, sound drivers, mouse drivers, disk caches, etc.

V86 monitor: a DOS device driver that transitions the 80386 (or higher) CPU from real mode to virtual 8086 mode (V86 mode). The program maintains control of V86 mode via its code and data that resides and is executed in the protected mode of the CPU.

[The list of terms used in describing the functionality of QEMM would be in alphabetical order and should include all technical terms used in the FS.]

1.3 References

This subsection should

a) Provide a complete list of all documents referenced elsewhere in the FS
b) Identify each document by title, report number (if applicable), date, and publishing organization
c) Specify the sources from which the references can be obtained

This information may be provided by reference to an appendix or to another document.

---

1. Introduction The introduction of the FS should provide an overview of the entire FS. It should contain the

following subsections

a) Purpose b) Definitions, acronyms, and abbreviations c) References

1.1 Purpose This subsection should

a) Describe briefly the purpose of the software b) Identify the intended audience for the software

Example:

QEMM 7.53 provides the user with as much conventional memory as possible for running programs while also providing all of the published memory management services used by DOS programs and devices. The product also contains features to display the current DOS and Windows memory configurations to help the end-user identify problems and opportunities for improving the system.

QEMM 7.53 is intended for retail distribution into the DOS and MS Windows utility market. It is expected to be an attractive product to the power user, the computer game player, and the network and multimedia users who need extra conventional memory to run their DOS programs.

1.2 Definitions, acronyms, and abbreviations This subsection should provide the definitions of all terms, acronyms, and abbreviations required to interpret properly the FS. This information may be provided by reference to one or more appendixes in the FS or by reference to other documents.

Example:

CPU: the computer chip that is the Central Processing Unit of the PC. For example, the Intel 80386SX, 80386DX, i486SX, i486DX, or Pentium processors and the compatible processors made by AMD, Cyrix, and others.

TSR: a “Terminate and Stay Resident” program, typically loaded in the AUTOEXEC.BAT file or other batch file executed during the boot sequence, which loads into memory, but then terminates back to the command processor prompt, leaving a portion of itself resident in memory to provide some service to the user. Examples of TSRs are network drivers, sound drivers, mouse drivers, disk caches, etc.

V86 monitor: a DOS device driver that transitions the 80386 (or higher) CPU from real mode to virtual 8086 mode (V86 mode). The program maintains control of V86 mode via its code and data that resides and is executed in the protected mode of the CPU.

[The list of terms used in describing the functionality of QEMM would be in alphabetical order and should include all technical terms used in the FS.]

1.3 References This subsection should

a) Provide a complete list of all documents referenced elsewhere in the FS b) Identify each document by title, report number (if applicable), date, and publishing organization c) Specify the sources from which the references can be obtained This information may be provided by reference to an appendix or to another document.

Example:

The MRD for the product, QEMM 7.53, is available [give location].

The V86 monitor in the product will provide industry-standard memory services described in publicly available documents. Following is a list of the documents that describe the memory services:

a) *Expanded Memory Specification 4.0 (EMS 4.0)*: EMS 4.0 was published in August, 1987 by Lotus Development Corporation, Intel Corporation, and Microsoft Corporation as document 300275-004 and can be obtained by [give source]. This specification was designed to allow DOS programs (whether applications or device drivers or TSRs) to ask an Expanded Memory Manager (EMM) to map pages from a (potentially 32 MB) expanded memory pool into a given place in the first megabyte of address space. This allows applications to store more than 640K of their code and data in expanded memory and access that part of their programs in a manner which will be substantially faster than reading it in from disk.

b) *Virtual Control Program Interface (VCPI)*: VCPI was published in [give year] by Quarterdeck and PharLap and can be obtained by [give source]. [give explanation of the purpose of VCPI]

c) *Virtual DMA Services*: The VDS specification is available [give location]. The purpose of VDS is to provide mechanisms for programs to find out the relationship between the linear address space and the physical memory that is mapped into that address space and to control the V86 monitor's control of those addresses. For example, programs that do DMA (Direct Memory Access) must give the address of the physical memory they are accessing, not the linear address, to the device with which they interact.

[The list of references used in describing the memory management services provided by QEMM would be in alphabetical order and should include all public specifications and documents.]

Example:

The MRD for the product, QEMM 7.53, is available [give location].

The V86 monitor in the product will provide industry-standard memory services described in publicly available documents. Following is a list of the documents that describe the memory services:

a) Expanded Memory Specification 4.0 (EMS 4.0): EMS 4.0 was published in August, 1987 by Lotus

Development Corporation, Intel Corporation, and Microsoft Corporation as document 300275-004 and can be obtained by [give source]. This specification was designed to allow DOS programs (whether applications or device drivers or TSRs) to ask an Expanded Memory Manager (EMM) to map pages from a (potentially 32 MB) expanded memory pool into a given place in the first megabyte of address space. This allows applications to store more than 640K of their code and data in expanded memory and access that part of their programs in a manner which will be substantially faster than reading it in from disk. b) Virtual Control Program Interface (VCPI): VCPI was published in [give year] by Quarterdeck and

PharLap and can be obtained by [give source]. [give explanation of the purpose of VCPI] c) Virtual DMA Services: The VDS specification is available [give location]. The purpose of VDS is to provide mechanisms for programs to find out the relationship between the linear address space and the physical memory that is mapped into that address space and to control the V86 monitor's control of those addresses. For example, programs that do DMA (Direct Memory Access) must give the address of the physical memory they are accessing, not the linear address, to the device with which they interact. [The list of references used in describing the memory management services provided by QEMM would be in alphabetical order and should include all public specifications and documents.]

2. Overall description

This section of the FS should describe the general factors that affect the product and its requirements. This section does not state specific requirements. Instead, it provides a background for those requirements, which are defined in detail in section 3, and makes them easier to understand.

This section usually consists of four subsections

a) Product perspective
b) Product components
c) Product constraints
d) Proposed future requirements

Example:

QEMM is a utility program for the PC which works behind the scenes to provide as much memory as possible for running programs. Not only do today's computers have more power and memory, but PC programs are larger and make greater demands on memory than ever before. A typical PC may be simultaneously running a multitasking environment such as DESQview or Microsoft Windows, a disk compression utility such as Stacker or DoubleSpace, multiple TSRs (such as anti-virus programs) and device drivers for a network, mouse, CD ROM, sound board, and other peripheral devices. And, different programs use different kinds of memory: conventional memory, upper memory, expanded memory, and extended memory. QEMM manages all these kinds of memory and can automatically transform the PC's memory into whatever kind of memory a program needs.

2.1 Product perspective

This subsection of the FS should put the product into perspective with other related products. If the product is independent and totally self-contained, it should be so stated here. If the FS defines a product that is a component of a larger system, as frequently occurs, then this subsection should relate the requirements of that larger system to functionality of the software and should identify interfaces between that system and the software.

Example:

QEMM 7.53 will be sold in the retail channel as a stand-alone product. It will also be bundled with a number of other Quarterdeck products, namely DESQview-386, DESQview/X, and Game Runner.

The bundling of QEMM with DESQview-386 and DESQview/X is designed to ensure that those multi-tasking environments have access to the V86 monitoring and memory mapping services that QEMM provides. In particular, some custom programming interfaces between the DESQview kernel and QEMM shall exist to allow DESQview to

a) Virtualize the screen
b) Use its "protection level" feature
c) support hardware interrupt reflection to a specific set of addresses independent of the contents of the real mode interrupt vector table.

The QEMM product bundled with DESQview-386 and DESQview/X will be identical to that sold separately, with the exception of the installation program. The installation programs for DESQview-386 and DESQview/X will be responsible for installing QEMM and the multitasking product together.

The QEMM product bundled with Game Runner will be identical to that sold separately, with the exception that the V86 monitor will not include the QuickBoot feature, the MS Windows versions of QSETUP and Manifest will not be included, and the installation program will be different.

QEMM 7.53 competes against other commercial memory managers and with EMM386, which ships with DOS 5 and later.

---

2. Overall description This section of the FS should describe the general factors that affect the product and its

requirements. This section does not state specific requirements. Instead, it provides a background for those requirements, which are defined in detail in section 3, and makes them easier to understand.

This section usually consists of four subsections

a) Product perspective b) Product components c) Product constraints d) Proposed future requirements

Example:

QEMM is a utility program for the PC which works behind the scenes to provide as much memory as possible for running programs. Not only do today's computers have more power and memory, but PC programs are larger and make greater demands on memory than ever before. A typical PC may be simultaneously running a multitasking environment such as DESQview or Microsoft Windows, a disk compression utility such as Stacker or DoubleSpace, multiple TSRs (such as anti-virus programs) and device drivers for a network, mouse, CD ROM, sound board, and other peripheral devices. And, different programs use different kinds of memory: conventional memory, upper memory, expanded memory, and extended memory. QEMM manages all these kinds of memory and can automatically transform the PC's memory into whatever kind of memory a program needs.

2.1 Product perspective This subsection of the FS should put the product into perspective with other related products. If the product is independent and totally self-contained, it should be so stated here. If the FS defines a product that is a component of a larger system, as frequently occurs, then this subsection should relate the requirements of that larger system to functionality of the software and should identify interfaces between that system and the software.

Example:

QEMM 7.53 will be sold in the retail channel as a stand-alone product. It will also be bundled with a number of other Quarterdeck products, namely DESQview-386, DESQview/X, and Game Runner.

The bundling of QEMM with DESQview-386 and DESQview/X is designed to ensure that those multi- tasking environments have access to the V86 monitoring and memory mapping services that QEMM provides. In particular, some custom programming interfaces between the DESQview kernel and QEMM shall exist to allow DESQview to

a) Virtualize the screen b) Use its "protection level" feature c) support hardware interrupt reflection to a specific set of addresses independent of the contents of

the real mode interrupt vector table. The QEMM product bundled with DESQview-386 and DESQview/X will be identical to that sold separately, with the exception of the installation program. The installation programs for DESQview-386 and DESQview/X will be responsible for installing QEMM and the multitasking product together.

The QEMM product bundled with Game Runner will be identical to that sold separately, with the exception that the V86 monitor will not include the QuickBoot feature, the MS Windows versions of QSETUP and Manifest will not be included, and the installation program will be different.

QEMM 7.53 competes against other commercial memory managers and with EMM386, which ships with DOS 5 and later.

2.1.1 User characteristics

This subsection of the FS should describe those general characteristics of the intended users of the product including educational level, experience, and technical expertise. It should not be used to state specific requirements but rather should provide the reasons why certain specific requirements are specified in later portions of the FS.

2.2 Product components

This subsection of the FS should provide a summary of the major subsystems included in the software. For example, a suite of utilities would list most or all of the executables in the suite in this section. An application program would list those components of the application that might warrant a separate section in the user manual. The components listed in this section should be described without mentioning the vast amount of detail that each of those functions requires.

If a detailed MRD exists for the product, then most of this section of the FS can be taken directly from the relevant section of the MRD. The components should be organized and described in a way that makes the list understandable to anyone unfamiliar with the product. Textual or graphical methods can be used to show the different functions and their relationships. Such a diagram is not intended to show a design of a product but simply shows the logical relationships among components.

The component list should include a descriptive name for the component, and the component should be referred to by that name for the rest of the document.

The "Specific Requirements" section of the FS should be organized in a way to match the list of components in this section. It is important to choose the components and the order of the components carefully, to ensure that the "Specific Requirements" section is as clear as possible.

Example:

The QEMM product consists of the following components:

a) *V86 monitor*: a device driver providing various memory services;
b) *DOS resource supplementers*: a set of utilities for increasing DOS resources (BUFFERS, FCBS, FILES, LASTDRIVE) after DOS has allocated the resources specified by the user in the CONFIG.SYS;
c) *Device driver converter*: a utility to load device drivers as if they were TSRs, to allow loading of device drivers from the shell command line;
d) *Resident program loader*: a loader program which can load TSRs and device drivers into memory regions specified by configuration parameters;
e) *Configuration optimizer*: a program to automatically identify the optimal loading locations for a user's device drivers and TSRs and will automatically configure the user's V86 monitor parameters and boot files (CONFIG.SYS, AUTOEXEC.BAT and called .BAT files) to maximize the available conventional memory while ensuring that the system functions;

[the list of components would continue here]

2.3 Product constraints

This subsection should also describe how the software operates inside various constraints. For example, these could include:

a) User interfaces
b) Hardware interfaces
c) Software interfaces
d) Memory constraints
e) Other constraints
f) Assumptions and dependencies

2.1.1 User characteristics This subsection of the FS should describe those general characteristics of the intended

users of the product including educational level, experience, and technical expertise. It should not be used to state specific requirements but rather should provide the reasons why certain specific requirements are specified in later portions of the FS.

2.2 Product components This subsection of the FS should provide a summary of the major subsystems included in the software. For example, a suite of utilities would list most or all of the executables in the suite in this section. An application program would list those components of the application that might warrant a separate section in the user manual. The components listed in this section should be described without mentioning the vast amount of detail that each of those functions requires.

If a detailed MRD exists for the product, then most of this section of the FS can be taken directly from the relevant section of the MRD. The components should be organized and described in a way that makes the list understandable to anyone unfamiliar with the product. Textual or graphical methods can be used to show the different functions and their relationships. Such a diagram is not intended to show a design of a product but simply shows the logical relationships among components.

The component list should include a descriptive name for the component, and the component should be referred to by that name for the rest of the document.

The “Specific Requirements” section of the FS should be organized in a way to match the list of components in this section. It is important to choose the components and the order of the comp onents carefully, to ensure that the “Specific Requirements” section is as clear as possible.

Example:

The QEMM product consists of the following components:

a) V86 monitor: a device driver providing various memory services; b) DOS resource supplementers: a set of utilities for increasing DOS resources (BUFFERS, FCBS,

FILES, LASTDRIVE) after DOS has allocated the resources specified by the user in the CONFIG.SYS; c) Device driver converter: a utility to load device drivers as if they were TSRs, to allow loading of

device drivers from the shell command line; d) Resident program loader: a loader program which can load TSRs and device drivers into memory

regions specified by configuration parameters; e) Configuration optimizer: a program to automatically identify the optimal loading locations for a

user's device drivers and TSRs and will automatically configure the user's V86 monitor parameters and boot files (CONFIG.SYS, AUTOEXEC.BAT and called .BAT files) to maximize the available conventional memory while ensuring that the system functions; [the list of components would continue here]

2.3 Product constraints This subsection should also describe how the software operates inside various constraints. For example, these could include:

a) User interfaces b) Hardware interfaces c) Software interfaces d) Memory constraints e) Other constraints f) Assumptions and dependencies

2.3.1 User interfaces

This should specify the following:

a) The common characteristics of the user interface components of the product. This includes those configuration characteristics (e.g., menu layout, colors, key usage and mouse usage; dialog box layout, colors, key usage and mouse usage). Reference to one or more style guides might be appropriate here.
b) All the aspects of optimizing the interface with the person who must use the system. This may simply comprise a list of do's and don'ts on how the system will appear to the user. One example may be a requirement for the option of long or short error messages. Like all others, these requirements should be verifiable, for example, "When the program is running on a 20mhz 386SX with 4 MB RAM and no tasks other than Program Manager running, any menu will be completely within 0.3 seconds of the user clicking on the menu bar" rather than "menus should be responsive." (This may also be specified in the Software System Attributes under a section titled Ease of Use.)

Example:

The QEMM 7.53 contains a number of programs which are device drivers or TSR's that do not require user input other than command line switches. The parsing of the command line switches for all programs in the product should be consistent. The output from each of these programs should be redirectable to DOS files when possible.

The product consists of three programs which have both DOS and MS Windows implementations: Install, Setup, and Manifest. When possible, both implementations of each program should have the same keystroke and mouse actions to activate a command. However, the look of the output from these programs should be appropriate for the respective environment, and should not be restricted by the need to have two implementations. Any user interface components included in the product that look like (or are) standard components of the MS Windows interface should comply with the April 1995 Microsoft Developers Network Product Documentation / SDKs / The Windows Interface Guidelines on-line document. Other sources of style guides are The Windows Interface - An Application Design Guide from Microsoft Press and the Visual Design Guide on-line document shipped with Visual Basic 3.0 from Microsoft.

2.3.2 Hardware interfaces

This should specify the logical characteristics of each interface between the software product and the hardware components of the system. This includes configuration characteristics (number of ports, instruction sets, etc.) It also covers such matters as what devices are to be supported, how they are to be supported, and protocols.

Example:

QEMM 7.53 is a utility designed exclusively for PC platforms with CPUs which are compatible with or a superset of the Intel 386SX and at least 1 MB of extended memory. The software be compatible with all such computers, including:

a) Diskless PCs
b) PCs with monochrome video adapters, CGA video adapters, or EGA/VGA compatible video adapters

Any information gathering algorithms used by the product's Manifest program that rely on hardware interfaces must work reliably regardless of whether the hardware is installed.

2.3.3 Software interfaces

This should specify the use of other required software products (for example, an operating system, a software library, a network stack, or other device driver), and interfaces with other application systems (for example, DDE or OLE). For each interface supplied, supported or used by the product, the following should be provided:

2.3.1 User interfaces This should specify the following:

a) The common characteristics of the user interface components of the product. This includes those

configuration characteristics (e.g., menu layout, colors, key usage and mouse usage; dialog box layout, colors, key usage and mouse usage). Reference to one or more style guides might be appropriate here. b) All the aspects of optimizing the interface with the person who must use the system. This may

simply comprise a list of do's and don'ts on how the system will appear to the user. One example may be a requirement for the option of long or short error messages. Like all others, these requirements should be verifiable, for example, "When the program is running on a 20mhz 386SX with 4 MB RAM and no tasks other than Program Manager running, any menu will be completely within 0.3 seconds of the user clicking on the menu bar" rather than "menus should be responsive." (This may also be specified in the Software System Attributes under a section titled Ease of Use.) Example:

The QEMM 7.53 contains a number of programs which are device drivers or TSR's that do not require user input other than command line switches. The parsing of the command line switches for all programs in the product should be consistent. The output from each of these programs should be redirectable to DOS files when possible.

The product consists of three programs which have both DOS and MS Windows implementations: Install, Setup, and Manifest. When possible, both implementations of each program should have the same keystroke and mouse actions to activate a command. However, the look of the output from these programs should be appropriate for the respective environment, and should not be restricted by the need to have two implementations. Any user interface components included in the product that look like (or are) standard components of the MS Windows interface should comply with the April 1995 Microsoft Developers Network Product Documentation / SDKs / The Windows Interface Guidelines on-line document. Other sources of style guides are The Windows Interface - An Application Design Guide from Microsoft Press and the Visual Design Guide on-line document shipped with Visual Basic 3.0 from Microsoft.

2.3.2 Hardware interfaces This should specify the logical characteristics of each interface between the software product and the hardware components of the system. This includes configuration characteristics (number of ports, instruction sets, etc.) It also covers such matters as what devices are to be supported, how they are to be supported, and protocols.

Example:

QEMM 7.53 is a utility designed exclusively for PC platforms with CPUs which are compatible with or a superset of the Intel 386SX and at least 1 MB of extended memory. The software be compatible with all such computers, including:

a) Diskless PCs b) PCs with monochrome video adapters, CGA video adapters, or EGA/VGA compatible video

adapters Any information gathering algorithms used by the product's Manifest program that rely on hardware interfaces must work reliably regardless of whether the hardware is installed.

2.3.3 Software interfaces This should specify the use of other required software products (for example, an operating system, a software library, a network stack, or other device driver), and interfaces with other application systems (for example, DDE or OLE). For each interface supplied, supported or used by the product, the following should be provided:

a) Name
b) Mnemonic
c) Specification number
d) Version number
e) Source
f) Discussion of the purpose of the interfacing software as related to this software product.
g) Definition of the interface in terms of message content and format. It is not necessary to detail any well-documented interface, but a reference to the document defining the interface is required.

Example:

QEMM 7.53 requires DOS 3.1 or later. The QEMM 7.53 device drivers and TSRs are designed exclusively for DOS-compatible operating environments where a DOS driver can gain control of the computer's protected mode. This means that the memory management parts of the product need not be compatible with running in OS/2 or Windows NT or Unix or other such environment, except in that they should exit elegantly when run in such an environment. Also, the product need execute properly when run in a DOS environment where some other program (such as another memory manager) has already grabbed control of protected mode.

The Manifest software should run in any DOS-compatible operating environment, even those running within a non-DOS protected mode operating system. Manifest should provide accurate information in those environments. It is acceptable for Manifest to use unpublished or unofficial interfaces to gather its information, when necessary.

QEMM will support the following set of public software interfaces (for more detail on these interfaces, see the References section of this document, above):

a) *Expanded Memory Specification 4.0 (EMS 4.0)*
b) *Virtual Control Program Interface (VCPI)*
c) *Virtual DMA Services (VDS)*

[the complete list of specs would be included above, in alphabetical order]

2.3.4 Memory constraints

This should specify any applicable characteristics and limits on primary and secondary memory.

Example:

Each of the components of QEMM should be able to execute reliably when there is as little as 384K available conventional memory and no extended or expanded memory, which the following exceptions:

a) The QEMM device driver may require as much as 1 MB of available extended memory and 448K of available conventional memory.
b) The DPMI device driver may require as much as 640K of available extended memory.

2.3.5 Other constraints

This subsection of the FS should provide a general description of any other items that will limit the developer's options. These include

a) Regulatory policies
b) Hardware limitations (for example, signal timing requirements)
c) Interfaces to other applications
d) Parallel operation
e) Audit functions
f) Control functions
g) Higher-order language requirements
h) Signal handshake protocols (for example, XON-XOFF, ACK-NACK)
i) Reliability requirements
j) Criticality of the application

---

a) Name b) Mnemonic c) Specification number d) Version number e) Source f) Discussion of the purpose of the

interfacing software as related to this software product. g) Definition of the interface in terms of message content and format. It is not necessary to detail any

well-documented interface, but a reference to the document defining the interface is required. Example:

QEMM 7.53 requires DOS 3.1 or later. The QEMM 7.53 device drivers and TSRs are designed exclusively for DOS-compatible operating environments where a DOS driver can gain control of the computer's protected mode. This means that the memory management parts of the product need not be compatible with running in OS/2 or Windows NT or Unix or other such environment, except in that they should exit elegantly when run in such an environment. Also, the product need execute properly when run in a DOS environment where some other program (such as another memory manager) has already grabbed control of protected mode.

The Manifest software should run in any DOS-compatible operating environment, even those running within a non-DOS protected mode operating system. Manifest should provide accurate information in those environments. It is acceptable for Manifest to use unpublished or unofficial interfaces to gather its information, when necessary.

QEMM will support the following set of public software interfaces (for more detail on these interfaces, see the References section of this document, above):

a) Expanded Memory Specification 4.0 (EMS 4.0) b) Virtual Control Program Interface (VCPI) c) Virtual DMA Services (VDS) [the complete list of specs would be included above, in alphabetical order]

2.3.4 Memory constraints This should specify any applicable characteristics and limits on primary and secondary memory.

Example:

Each of the components of QEMM should be able to execute reliably when there is as little as 384K available conventional memory and no extended or expanded memory, which the following exceptions:

a) The QEMM device driver may require as much as 1 MB of available extended memory and 448K of

available conventional memory. b) The DPMI device driver may require as much as 640K of available extended memory.

2.3.5 Other constraints This subsection of the FS should provide a general description of any other items that will limit the developer's options. These include

a) Regulatory policies b) Hardware limitations (for example, signal timing requirements) c) Interfaces to other applications d) Parallel operation e) Audit functions f) Control functions g) Higher-order language requirements h) Signal handshake protocols (for example, XON-XOFF, ACK-NACK) i) Reliability requirements j) Criticality of the application

k) Safety and security considerations

Example:

QEMM 7.53 will be compatible with the WDEB386 debugger, which relies on certain hookouts to share the protected mode environment with QEMM.

2.3.6 Assumptions and dependencies

This subsection of the FS should list each of the factors that affect the requirements stated in the FS. These factors are not design constraints on the software but are, rather, any changes to them that can affect the requirements in the FS. For example, an assumption may be that a specific operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not available, the FS would then have to change accordingly.

Example:

Certain components of QEMM 7.53 contain assumptions and dependencies. Should any of these assumptions or dependencies prove incorrect for future versions of hardware and software, then the FS will have to change to address the problems. Known assumptions are

a) The *DOS kernel relocator* component assumes certain things about the initialization sequence and internal structure of the DOS kernel
b) The *DOS resources relocator* component assumes certain things about the structures internal to the DOS kernel which describe the DOS resources
c) The *DBLSPACE/DRVSPACE relocator* component assumes certain things about the initialization, structures and interfaces of the MS-DOS disk compression driver

[Myriad other assumptions in the QEMM product would follow]

2.4 Proposed future requirements

This subsection of the FS should identify requirements that may be delayed until future versions of the system.

k) Safety and security considerations Example:

QEMM 7.53 will be compatible with the WDEB386 debugger, which relies on certain hookouts to share the protected mode environment with QEMM.

2.3.6 Assumptions and dependencies This subsection of the FS should list each of the factors that affect the requirements stated in the FS. These factors are not design constraints on the software but are, rather, any changes to them that can affect the requirements in the FS. For example, an assumption may be that a specific operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not available, the FS would then have to change accordingly.

Example:

Certain components of QEMM 7.53 contain assumptions and dependencies. Should any of these assumptions or dependencies prove incorrect for future versions of hardware and software, then the FS will have to change to address the problems. Known assumptions are

a) The DOS kernel relocator component assumes certain things about the initialization sequence and

internal structure of the DOS kernel b) The DOS resources relocator component assumes certain things about the structures internal to

the DOS kernel which describe the DOS resources c) The DBLSPACE/DRVSPACE relocator component assumes certain things about the initialization,

structures and interfaces of the MS-DOS disk compression driver [Myriad other assumptions in the QEMM product would follow]

2.4 Proposed future requirements This subsection of the FS should identify requirements that may be delayed until future versions of the system.

3. Specific requirements

This section of the FS should contain all the software requirements to a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements. The contents of this section, when combined with a product prototype, should be sufficient to guide a technical writer to write the end-user documentation for the product.

Throughout this section, every stated requirement should be externally perceivable by users or should be demonstrable by other external systems. These requirements should include at a minimum a description of every input (stimulus) into the system, every output (response) from the system and all functions performed by the system in response to an input or in support of an output.

As this is often the largest and most important part of the FS, careful organization of the section is important. The goal would be for a user of the FS to be able to find quickly the specific requirements of a particular component. An important organizational guideline is to avoid nesting too far down in the outline structure. For example, you don't want to have the sub-component of a sub-component of a sub-component of a component of the screen layout described in the screen layout section. Doing so would make it difficult to identify the high level screen layout components (as there would be hundreds of low-level items described in between the high-level items).

Another good guideline is to only identify any component in one location. Even if a dialog box can be brought up by many different events, the specific requirements of the dialog box should only be described once. The events that bring up the dialog could be detailed in the descriptions of higher level components. For example, the File/Open dialog could be brought up by selecting the Open item from the File menu or by typing Ctrl+O in the document window. The File/Open dialog should be described in its own section, with cross-references to the sections describing the File menu and the document window.

The exact structure of this section will vary depending upon the nature of the product. There are likely to be three distinct types of entries that will be used in the section

a) A sequence of functions that are executed without user interaction and which create little user output except under error conditions (e.g. the program initialization sequence)
b) A screen layout component which is generally visible on the screen during program execution (e.g. a status bar or document window)
c) A user interface component that appears when some input, or one of various events, occurs (e.g. a particular dialog box or menu)

Based on the above, a possible organization for this section of the FS would be

3.1 Initialization sequence
- 3.1.1 Function #1 and potential error messages
- 3.1.2 Function #2 and potential error messages
- •
- 3.1.n Function #n and potential error messages

3.2 Screen layout
- 3.2.1 Application area
  - 3.2.1.1 Application area component #1
  - •
  - 3.2.1.n Application area component #2
- 3.2.2 Tool bar
  - 3.2.2.1 Tool bar component #1
  - •
  - 3.2.2.n Tool bar component #2
- 3.2.3 Menu bar
  - 3.2.3.1 Menu bar component #1
  - •

---

3. Specific requirements This section of the FS should contain all the software requirements to a level of detail

sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements. The contents of this section, when combined with a product prototype, should be sufficient to guide a technical writer to write the end-user documentation for the product.

Throughout this section, every stated requirement should be externally perceivable by users or should be demonstrable by other external systems. These requirements should include at a minimum a description of every input (stimulus) into the system, every output (response) from the system and all functions performed by the system in response to an input or in support of an output.

As this is often the largest and most important part of the FS, careful organization of the section is important. The goal would be for a user of the FS to be able to find quickly the specific requirements of a particular component. An important organizational guideline is to avoid nesting too far down in the outline structure. For example, you don't want to have the sub-component of a sub-component of a sub- component of a component of the screen layout described in the screen layout section. Doing so would make it difficult to identify the high level screen layout components (as there would be hundreds of low- level items described in between the high-level items).

Another good guideline is to only identify any component in one location. Even if a dialog box can be brought up by many different events, the specific requirements of the dialog box should only be described once. The events that bring up the dialog could be detailed in the descriptions of higher level components. For example, the File/Open dialog could be brought up by selecting the Open item from the File menu or by typing Ctrl+O in the document window. The File/Open dialog should be described in its own section, with cross-references to the sections describing the File menu and the document window.

The exact structure of this section will vary depending upon the nature of the product. There are likely to be three distinct types of entries that will be used in the section

a) A sequence of functions that are executed without user interaction and which create little user

output except under error conditions (e.g. the program initialization sequence) b) A screen layout component which is generally visible on the screen during program execution (e.g.

a status bar or document window) c) A user interface component that appears when some input, or one of various events, occurs (e.g. a

particular dialog box or menu)

Based on the above, a possible organization for this section of the FS would be

3.1 Initialization sequence

3.1.1 Function #1 and potential error messages 3.1.2 Function #2 and potential error messages

- 3.1.n Function #n and potential error messages 3.2 Screen layout

3.2.1 Application area

3.2.1.1 Application area component #1

- 3.2.1.n Application area component #2 3.2.2 Tool bar

3.2.2.1 Tool bar component #1

- 3.2.2.n Tool bar component #2 3.2.3 Menu bar

3.2.3.1 Menu bar component #1

•

3.2.3.n Menu bar component #2
3.2.4 Status line
3.2.4.1 Status line component #1
•
3.2.4.n Status line component #2
•
3.2.n Screen area #n and components
3.2.n.1 Screen area #n component #1
•
3.2.n.n Screen area #n component #2

3.3 Particular components

3.3.1 Dialog boxes
3.3.1.1 Dialog box #1
•
3.3.1.n Dialog box #n
3.3.2 Menu items
3.3.2.1 Menu item #1
•
3.3.2.n Menu item #n
3.3.3 Error messages
3.3.3.1 Error message #1
•
3.3.3.n Error message #n

3.4 Termination sequence

3.4.1 Function #1 and potential error messages
3.4.2 Function #2 and potential error messages
•
3.4.n Function #n and potential error messages

Each item in the Specific Requirements section should contain a picture of the item and details about the item, including both content and format as follows:

a) Name of item
b) Description of purpose
c) Events that can invoke this item
d) Items that can be invoked by this item
e) Valid responses to the item
f) Effects of invalid responses to the item

Items that are inappropriate to the particular component should not be included in the description.

Example:

3.1 Initialization sequence

3.1.1 Copyright message

QEMM386.SYS displays its copyright message, for purposes of protecting the company's copyright:

Quarterdeck Expanded Memory Manager 386 V7.5
Copyright (c) 1986-1995 by Quarterdeck Corporation
U.S. Patent Numbers 5,237,669 and 5,367,658
Serial number: 123-QDK-12345-6789

Registered to: For QDK Internal Use Only !!!!

3.2.4.1 Status line component #1

- 3.2.4.n Status line component #2
- 3.2.n Screen area #n and components

3.2.n.1 Screen area #n component #1

- 3.2.n.n Screen area #n component #2 3.3 Particular components

3.3.1 Dialog boxes

3.3.1.1 Dialog box #1

- 3.3.1.n Dialog box #n 3.3.2 Menu items

3.3.2.1 Menu item #1

- 3.3.2.n Menu item #n 3.3.3 Error messages

3.3.3.1 Error message #1

- **3.3.3.n Error message #n 3.4 Termination sequence**

3.4.1 Function #1 and potential error messages 3.4.2 Function #2 and potential error messages

- 3.4.n Function #n and potential error messages

Each item in the Specific Requirements section should contain a picture of the item and details about the item, including both content and format as follows:

a) Name of item b) Description of purpose c) Events that can invoke this item d) Items that can be invoked by this item e) Valid responses to the item f) Effects of invalid responses to the item Items that are inappropriate to the particular component should not be included in the description.

Example:

3.1 Initialization sequence

3.1.1 Copyright message QEMM386.SYS displays its copyright message, for purposes of protecting the company's copyright:

Quarterdeck Expanded Memory Manager 386 V7.5 Copyright (c) 1986-1995 by Quarterdeck Corporation U.S. Patent Numbers 5,237,669 and 5,367,658 Serial number: 123-QDK-12345-6789

Registered to: For QDK Internal Use Only !!!!