DATE: 31 January 19

SOFTWARE SPECIFICATION

for

KA630-A CONSOLE PROGRAM

REVISION 2.2

Written By:	A. Randall Heap	MLO5-5/E71
Issued By:	A. Randall Heap	
Reviewed By:	Jesse Lipcon	MLO5-5/E71
	Jay Nichols	MLO5-5/E71
	Kathy Morse	ZK01-1/D42

C O M P A N Y C O N F I D E N T I A L

DATE: 31 January 19

SOFTWARE SPECIFICATION

for

KA630-A CONSOLE PROGRAM

REVISION 2.2

Written By:	A. Randall Heap	MLO5-5/E71
Issued By:	A. Randall Heap	
Reviewed By:	Jesse Lipcon	MLO5-5/E71
	Jay Nichols	MLO5-5/E71
	Kathy Morse	ZK01-1/D42

C O M P A N Y C O N F I D E N T I A L

PREFACE

This is a Software Product Specification. It is the definitive description in measurable terms of the goals, capabilities, and external characteristics of the KA630 console program. It is the commitment of WHAT IS TO BE BUILT as agreed by the Product Team.

No changes are to be made to the product as described here, unless approved and documented as an amendment to this Specification.

Associated Documents:

1. The KA630 Processor Specification which describes the processor board of which this software product (in ROM) is a component.

2. The KA630 System Development Plan, which describes the development project to design, build, test, evaluate, and deliver this product.

3. The KA630 ROM Diagnostic Specification which describes the ROM based diagnostics which are part of this program.

4. DEC Standard 32, the VAX Architecture Standard, which includes the definition of the VAX console system (13-Sep-82 - Rev 3).

5. The VIDEO SYSTEM STANDARDS Reference Manual which defines the video terminal architecture for DEC terminals.

6. DEC Standard 138, "Registry of Control Functions for Character Imaging Devices".

Change History:

Date: 25-July-1983

Revision 1.0 Original KA630 phase zero version.

Date: 22-February-1984

Revision 1.15 Revisions for phase 1
VMB details and figures defined according to "MicroVAX I CPU Technical Description", second draft, October 1983. This revision brings the specification up to phase 1 status. All subsequent revisions will be tracked by change bars in the left margin.

Revision 1.16 Revisions for phase 1 - continued
Spelling corrections, deleted vestigial text fragments

remaining from previous versions, a correction to th discussion of national replacement characters was made powerup mode was simplified, QVSS location wa described, boot flag bits 0 and 7 were dropped.

Revision 1.17 Revisions for phase 1 - continued

Power-up modes redefined, section on diagnostic enlarged, number of console languages enlarged from 5 t 10, documentation of all TOY registers strictly interna to the console program was removed, console flag bit moved to CPMBX TOY register, CPFLG TOY registe eliminated, power-up figures revised.

Revision 1.18 Revisions for phase 1 - continued

Revision of appendix a, LED and console output codes halt message texts converted to abbreviation so as t avoid requirements for translation.

Date: 22-Jun-1984

Revision 1.19 Revisions for delayed phase 1 - continued

Discussion of the power-up sequence and the overal organization of the beginning of the Softwar Capabilities section was revised. Wording around dis boot was modified.

Cleanup for a pre-phase-1 release.

Date: 18-Jul-1984

Revision 1.20 Converted processor name from KDQ32 to KA630, adde Portuguese to the list of supported languages.

Date: 2-Aug-1984

Revision 1.21 Changed spec name from KA630 to KA630-A. Delete vestigial documentation of "interrupt page" base console memory.

Date: 10-Sept-84

Revision 2.0 Text of user-friendly console messages revised an reduced. Post Phase 1 copy.

Date: 2-Nov-84

Revision 2.1 Rearrange test sequence, add QxSS information, chan
screen displays.

Date: 31 January 1985

Revision 2.2 Correct PROM boot definition. Correct language defau
alg. displays.

CONTENTS

TABLES

1 PRODUCT SUMMARY

This document describes the KA630 console program. This program conjunction with KA630 hardware implements the KA630 console syst which gains control whenever KA630 "halts". For the KA630, "haltin means only that control is transferred to this program, not that t processor stops executing instructions.

When the console program is running, it provides services expected of VAX-11 console system. In particular, the following services a available:

- Automatic restart or bootstrap following processor halts initial power-up.
- An interactive command language allowing the user to exami and alter the state of the processor.
- Diagnostic tests executed on power up that check out the CP the memory system and the Q22-bus map.
- Support of video or hardcopy terminals as the console termin as well as support of QVSS or QDSS based bitmapped terminals.

2 ENVIRONMENT

2.1 Users

Engineering, Manufacturing, Field Service and customers will use thi program to test, configure and boot their system. This program i needed by these users both to provide an easy means to bootstrap a KA63 system and to detect and isolate system malfunctions.

Of these users, all but customers are assumed to be compute "sophisticates". While this will often be true of customers as well, n such assumption is made. Target customers include both sophisticate users who can use and understand all the features provided by th console program as well as unsophisticated users who know next t nothing about computers.

Users are not assumed to speak English. The console program may b directed to output all console messages in either English, French German, Spanish, Italian, Norwegian, Dutch, Swedish, Finish, Danish c Portuguese. If when the system powers up there is no languag specified, it prompts the user for the language to use. The use language is recorded in battery backed up RAM so the preferred languag is retained when the system is shut off.

2.2 Hardware

The console program is located in ROM on the KA630 processor board. Th ROM address range is located in the KA630's local I/O space. Th program uses the KA630 processor LEDs and console terminal output t communicate diagnostic progress and error reports to the user.

A console terminal is not required for operation but halts should not b enabled on a system without a console terminal. A QxSS (either QVSS o QDSS) based display can be substituted for the console terminal.

In order for the console program to operate, the processor must b functioning to the point of executing instructions from the consol program ROM.

The KA630 decodes the ROM addresses so that the same ROM appears mor than once in the address space. The console program is written i position independent code so that it may be executed from any addres range and the KA630 uses this feature to selectively enable and disabl the external halt circuitry. If the console program is executing in th range of 20040000 to 2004FFFF (hex), external halt conditions ar ignored. If the console program is executing in the range of 200500 to 2005FFFF (hex) external halt conditions are honored and the consol program may be "halted" (whereupon it will immediately reenter at it beginning to process the halt).

The console program normally executes from the first address range whil the diagnostics software normally executes from the second addres range.

For more information on the processor hardware, consult the KA63 Processor Specification. See also appendix B for a summary of th hardware registers used by the console program.

2.3 Software

None - the console program runs standalone, it uses no other softwar products.

2.4 Services

None - the console program requires no support services.

3 SOFTWARE CAPABILITIES

The console program is divided into six major components plus th diagnostics. These components and their responsibilities are documente

in the sections that follow. Diagnostics are documented in "KA630 RC Diagnostic Specification".

Note that throughout this document, the console program is ofte referred to simply as the "console".

The console program receives control whenever the processor halts whether becasue of an external halt signal, the execution of a hal instruction, a serious system error or because of power-up. On any hal condition, the processor switches to physical addressing, saves the PC PSL and the ISP internally, encodes and saves the cause of the halt in halt code and then branches to the start of the console program ROM Upon entry, the console program always outputs the hex value "E" to th console LED's to indicate that at least one instruction has bee executed. It then waits for power to stabilize by looping until BDR<1 is set.

When these common preliminaries are completed, the console progra checks to see if the halt is a power-up halt or a halt for some othe reason. If it is a power-up halt, it begins the power-up sequenc described in section 3.1. If it is not a power-up halt, control passe to the entry and dispatch code described in section 3.2.

3.1 Power-up

Upon power-up, the console automatically performs a variety of one tim operations. These operations initialize the processor and are neede only on power-up.

3.1.1 Power-up Mode -

On power-up, BDR<10:9> is interpreted as a power-up mode field. Th interpretation of the power-up mode field is shown in table 1. A discussed in the sections that follow, several power-up operations ar dependent on the power-up mode.

Table 1: Power-up Modes

Mode	Language Prompt	Diagnostics
0	Prompt for language only if TOY battery backup failed.	Run full diagnostics.
1	Prompt for language on every power-up.	Run full diagnostics.

Table 1 (cont.)

Mode	Language Prompt	Diagnostics
2	Set language to English.	Run console terminal loop-back tests.
3	Set language to English.	Run abbreviated diagnostics.

3.1.2 Power Stabilization And ROM Checksum Checks -

Following the application of power, the processor first perfor hardware initialization and then control is transfered to the consol program ROM code. The common console startup functions were describe previously in section 3. Following their completion, on power-up control passes to the code described here.

Beginning the power-up processing, the console code outputs the he value of "D" to the LED's indicating that the power stabilization wai is over. It then calculates a checksum of the console program ROM an checks it against the valid checksum stored in the ROM itself. If th computed checksum differs from the stored checksum, the code "hangs" i a loop. If the checksum is the same, the power-up code proceeds to th next step.

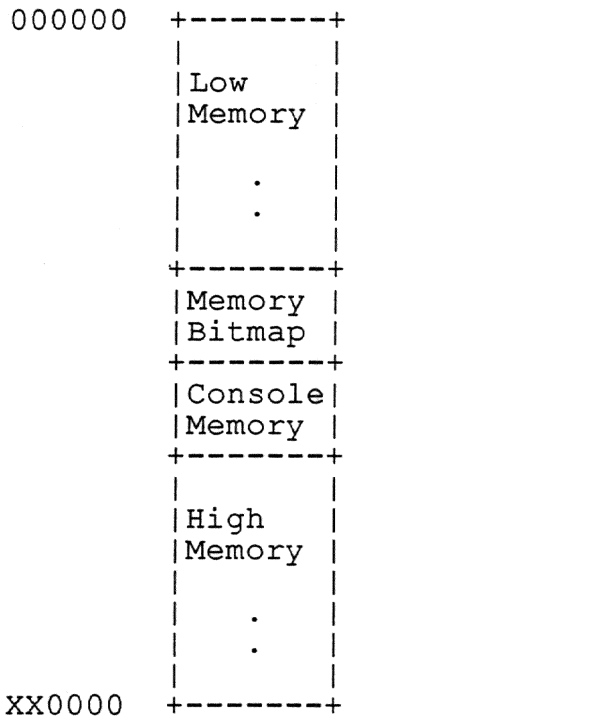
3.1.3 Console Program Initialization -

The next step of the power-up initialization is location an initialization of the memory needed for the console program itself. Th hex value of "C" is output to the LED's at the beginning of this step.

During this step, the console ROM code searches top down throug available memory for a small contiguous block that will be used by th console program for writeable storage. This block consists of two page for the console's direct use and additional pages for use to store bitmap of available memory. The amount of memory allocated to th bitmap varies according to the amount of memory available.

Following initialization, memory appears as shown in figure 1.

Figure 1: Console Memory Map



The console program memory is used by the console program for its stac and other data structures.

The bitmap is filled in later with a map of valid memory pages by th power-up memory diagnostics and this bitmap is passed to the bootstra as a map of valid memory. Beginning from the base of the bitmap, th first bit corresponds to the first page of low memory, the second bit t the second page and so on. If the bit is set, the page is good, if th bit is clear, the page failed the memory test. The bitmap does not ma itself nor any other memory that follows the bitmap. Since syste software is expected to use only pages marked as good in the bitmap, system software is not expected to modify the bitmap or the consol program memory. However, both the console program memory pages and th bitmap are checksumed by the console program to guard against accidenta modification by the system software.

If the console cannot locate enough memory for its own use and th bitmap, it "hangs".

Following initialization of its memory, the console program clears CPMBX<1:0> (halt action), CPMBX<2> (bootstrap in progress flag) an CPMBX<3> (restart in progress flag).

3.1.4 Battery Backup Check -

The next thing done by the power-up operations is to check the TOY cloc to see if the battery backup has failed. If this has happened, time c year has been lost along with the contents of all the TOY clock RAM. I the battery backup has failed, the console stops the TOY clock, zerc the time and all TOY RAM and then initializes the TOY clock control ar status registers. The operating system must check TOY clock control ar status register B and determine if the clock is stopped to know if th TOY clock contains a valid time.

No change is made to the LED's during this operation.

3.1.5 Test Interprocessor Communication Register -

Next, the interprocessor communication register is tested. The he value "B" is output to the LED's during this test.

This test is performed at this point as in addition to testing th register itself, this tests whether or not the Q22-bus is arbitratin properly. If it is not arbitrating, the processor will hang here Hanging here is preferable to hanging in the next step where we chec for QxSS video hardware on the bus. Were we to hang in the next ste without first checking the Q22-bus, the resulting LED display would b ambiguous as to which failed, the Q22-bus or the QxSS.

3.1.6 Determining The Console Terminal Type -

3.1.6.1 QxSS Determination -

Next, if the processor is an arbiter processor, the console progra checks for the presence or absence of a QxSS video display as th console device. Either a QVSS or a QDSS will be detected. If th processor is an auxiliary processor, this test is skipped. The he value "A" is output to the LED's during this test.

The QxSS is determined by testing for the presence of its CSR address a tbs. (Both QVSS and QDSS use the same CSR address.) If there is n response, no QxSS is assumed present and the console program moves t the console terminal determination code which follows. If a QxSS i detected, it is initialized and a short diagnostic is executed. If th initialization and diagnostics succeed, the console will use the QxSS a its console terminal (however, see appendix C) and skips the next ste and moves directly to the step described in section 3.1.7. If eithe the initialization or the diagnostic fails, the system hangs.

3.1.6.2 Console Terminal Determination -

If no QxSS is detected, it is assumed that normal a console terminal i connected to the console port or that NO terminal is connected. Th console program then next attempts to determined the type of termina connected. The value of "9" is output to the LED's during this test.

This determination is done by sending the terminal a device attribut request escape sequence. If the device responds with a recognizabl response, the terminal is classified as a video terminal versus a har copy terminal based on that response. This information is used when i console I/O mode to govern how command line editing is performed. Th terminal must respond in 1 second to the device query. If no respons or the response is not recognized, the test is repeated two more time or until the response is understood. If the device never responds c the response isn't recognized, the terminal is classified as a hard cop terminal.

Terminal response will be recognized in either eight bit or seven bi mode.

In addition to determining the presence or absence of a recognizable CR console, this information is also used to determine if the termina supports the DEC multinational character set (MCS). The console progra assumes that all new terminals, VT2xx and beyond support MCS. If th terminal does not support MCS, CPMBX<7:4> is set to 2, forcing Englis as the console display language.

3.1.7 Console Message Language Check -

The console next outputs a hex value of "8" to the LED's and the determines the appropriate language to use for all console messages The algorithim used to determine the language appears below. Th console language is stored in CPMBX<7:4> (see appendix B).

1. If power-up mode (see table 1) is 2 or 3, set the consol language to English and exit.

2. If power-up mode is 0 or if the value of CPMBX<7:4> is zero solicit the language from the user. If the user doesn' respond within 30 seconds, set the language to English (3) an exit.

Remember that when the terminal is queried, if it isn't recognized a one that supports MCS, CPMBX<7:4> is set to 2, forcing English as th console language. English messages use the 7 bit subset of MCS. Also, if a loss of power for the TOY clock is detected, the contents of th TOY RAM is zeroed meaning that step 2 of the algorithm above will caus the user to be prompted.

The format of the user query issued in step 2 is shown in Figure 2.

Figure 2: User Language Prompt

```
    1) Danish              7) Dutch
    2) German              8) Norwegian
    3) English             9) Portuguese
    4) Spanish            10) Finish
    5) French             11) Swedish
    6) Italian

  (1-11):
```

The text will be displayed using the DEC Multinational Character se extension.

The user must respond by entering a value of 1 through 11 following th question mark prompt, any other input will be rejected and the promp line will be issued again. Normal console input line editing feature (see section 3.6.1) will be available during input.

If the console program has determined that a QxSS video display syste is being used as the console, an additional step is required. The QxS display system uses the DEC LK201 keyboard which comes in 16 nationa variants. The keyboard variant cannot be determined by querying th keyboard itself, it must be determined either from the language selecte or by means of an additiona menu selection. If French, German o English is specified as the language, the keyboard variant is ambiguou and the appropriate menu from figure 3 is displayed and the user i prompted to specify the which national keyboard variant is in use. I the user does not respond in 30 seconds, the last selection is assumed.

Figure 3: QxSS Keyboard prompts

```
----------French----------

      1) Canada
      2) France/Belgium
      3) Switzerland

      (1..3):

----------German----------

      1) Germany/Austria
      2) Switzerland

      (1..2):

----------English----------

      1) United Kingdom
      2) United States/Canada

      (1..2):
```

3.2 Entry/Dispatch

Following the determination of the console language on power-up, c directly on entry from any other halt condition, the console dispatche to the appropriate code to service the halt.

To determine what actions to take, the console program examines the ha code, the halt enable bit (BDR<14>), the console program mailbo register (CPMBX) and then acts in accordance with decision table show in table 2.

Table 2: Console Entry Decision Table

BDR(14)	Power-up	CPMBX(1:0)	Actions
F	T	X	Diagnostics, bootstrap, halt.
F	F	0	Restart, bootstrap, halt
F	F	1	Restart, halt.
F	F	2	Bootstrap, halt.
F	F	3	Halt
T	T	X	Diagnostics, halt.
T	F	0	Halt.
T	F	1	Restart, halt.
T	F	2	Bootstrap, halt.
T	F	3	Halt.

A power-up entry is one where the processor halt code is 3. "X" mean "don't care" and the meanings of the other CPMBX codes are defined i appendix B, section B.2.1.

Multiple actions in table 1 mean that the first action is taken and i and only if it fails, the next action is taken. Diagnostics are a exception. If diagnostics fail the console will "hang" witho attempting to bootstrap the processor. If they succeed, then the nex action is taken.

Note that because the KA630 does not support battery backup, it examine the halt code and does not attempt to perform restart operation following power-up.

3.3 Diagnostics

On power-up, the Entry/Dispatch code dispatches first to the diagnostic to check out the processor and memory before proceeding. Before doin this, the console outputs the message "Performing normal system tests. to the console terminal. As each test is run, a code is displayed o the processor LED's and output to the console terminal as well, causin a "countdown" to be displayed on both.

The first diagnostic LED code is 8 so executing the diagnostic continues the LED countdown. The diagnostic codes are documented i table A-1 along with all other codes. See figures 9 and 10 in section for console display formats.

At the conclusion of all tests, the message "Tests completed." is outpu to the console terminal.

If a diagnostic test detects a fatal error, an error message i displayed on the console, a summary message is displayed indicating tha the continued operation is not possible and then the system "hangs" leaving the test code in the LED's. If halts are disabled, the only wa

to clear the hung system is to turn the system off and then on agai If halts are enabled, the hung system may be cleared by manually halti it, causing it to enter console command mode.

Diagnostic software is fully documented in "KA630 ROM Diagnosti Specification".

3.4 Restart

The console can restart a halted operating system. To do so, th console searches system memory for the Restart Parameter Block (RPB), data structure constructed for this purpose by the operating system. I a valid RPB is found, the console restarts the operating system at a address specified in the RPB.

The console keeps a "restart in progress" flag in the CPMBX registe which it uses to avoid repeated attempts to restart a failing operatin system. An additional "restart in progress" flag may be maintained b software in the RPB.

The console uses the following algorithm to restart the operatin system:

1. Check to see if the "restart in progress" flag in CPMBX is set If so, restart fails.

2. Print the message "Restarting the operating system." on th console terminal.

3. Set the restart in progress flag.

4. Look for a Restart Parameter Block (RPB), left in memory by th operating system. If none is found, restart fails.

5. Read the software restart in progress flag from bit<0> of th fourth longword of the RPB. If it is set, restart fails.

6. Load SP with the address of the RPB plus 512.

7. Load AP with the halt code.

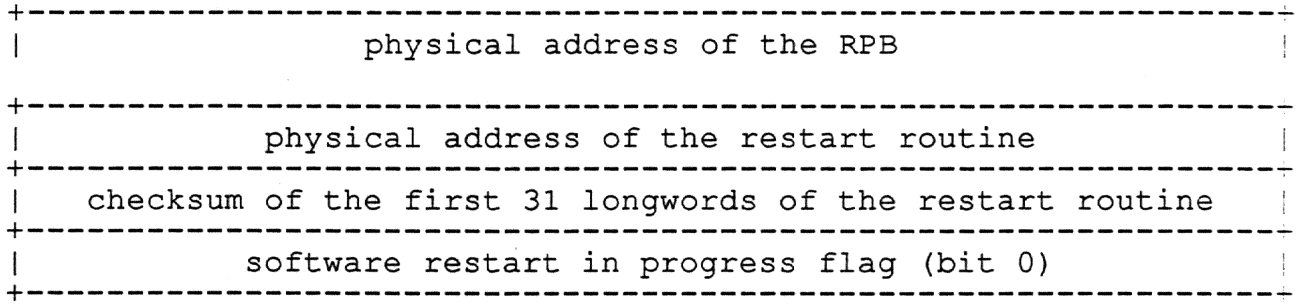
8. Display "0" in the console LED's.

9. Start the processor at the restart address, which is read fro the second longword in the RPB.

If restart fails, the console prints "Attempt to restart operatin system failed." on the console terminal. If the restart is successful the operating system must clear the restart in progress flag in CPMB (see appendix B for information on the CPMBX register).

The Restart Parameter Block is a page aligned control block, created b
the operating system. Its format is shown in figure 4.

Figure 4: RPB Format



Restart Parameter Block
(RPB)

The console uses the following algorithm to find a Restart Parameter
Block:

1. Search for a page of memory that contains its address in th
first longword. If none is found, the search for an RPB ha
failed.

2. Read the second longword in the page (the physical address o
the restart routine). If it is not a valid physical address
or if it is zero, return to step 1. The check for zero i
necessary to ensure that a page of zeros does not pass the tes
for a valid RPB.

3. Calculate the 32 bit twos-complement sum (ignoring overflows
of the first 31 longwords of the restart routine. If the su
does not match the third longword of the RPB, return to step 1

4. A valid RPB has been found.

The same algorithm is used for both arbiter and auxiliary processors.

3.5 Bootstrap

The console can load and start (bootstrap) an operating system. To d
so, it searches for a 64 kilobyte segment of correctly functionin
system memory, sets SP equal to the base address of the segment plus 51
and then copies the primary bootstrap, called VMB, from the consol
program ROM to the segment starting at the location specified by SP.
The console program then branches to the first location in VMB whic
then loads and starts the operating system.

To prevent a situation from occurring in which the console repeatedl tries and fails to bootstrap the operating system, it maintains "bootstrap in progress" flag in CPMBX. If a system bootstrap woul occur automatically but the flag is already set, the console assume that an attempt has already been made and has failed, so it does not tr again.

The console uses the following algorithm to bootstrap the operatin system:

1. If this bootstrap is the result of a console BOOT command, ski to step 4.

2. Check to see if the "bootstrap in progress" flag is set. I so, the bootstrap fails.

3. Print the message "Starting the operating system." on consol terminal.

4. Set the bootstrap in progress flag.

5. Locate a page-aligned 64 kilobyte segment of good memory. I such a segment cannot be found, the bootstrap fails.

6. Initialize the Q22-bus I/O map as described below.

7. Load the general registers for the primary bootstrap progra (VMB) as shown in table 3.

8. Copy VMB from the console ROM to 512 bytes past the base of th good segment.

9. Invoke VMB. If VMB fails, the bootstrap fails.

If bootstrap fails, the console prints "Attempt to start operatin system failed." on the console terminal.

If the bootstrap is successful, the operating system must clear th bootstrap and restart in progress flags in the CPMBX register and clea the LED display by depositing a value of 0 in the BDR register (se appendix B).

Table 3: VMB Register Usage

Rn	Description
R0	ASCII device name (from BOOT command) or zero.
R1	Contents of KA630 Boot and Diagnostic Register
R2	Memory bitmap size in bytes.
R3	Address of memory bitmap.
R4	Unused.
R5	Software boot control flags (from BOOT command only).
R10	halt PC value.
R11	halt PSL value.
AP	halt code.
SP	512 bytes past base of 64 Kb of good memory.

During step 6 of the bootstrap process the Q22-bus I/O map is initialized using the algorithm shown below. The main function of this initialization is to preset the arbiter processor I/O map so that all unoccupied pages of the Q22-bus are mapped to the corresponding pages in the first 4 Mb of local memory. (This is a MicroVAX I compatibility feature). Note that this is not done for auxiliary processors, any auxiliary Q22-bus I/O mapping must be coordinated with all other processors so all auxiliary processor I/O map registers are marked invalid.

1. Turn on IPCR<8>, the halt flag.

2. Disable the I/O map by clearing IPCR<5>.

3. If the arbiter processor, do the following for each I/O map register:

   a. Set the map register address bits to map the Q22-bus page to the corresponding local memory page.
   b. If the corresponding Q22-bus page is unoccupied, turn on the valid bit.
   c. If the page is occupied, turn off the valid bit.

   If an auxiliary processor, turn off the valid bit in all map registers.

4. Enable the I/O map by setting IPCR<5>.

5. If an auxiliary processor, loop until the IPCR<8> bit i cleared.

Steps 1 and 5 are present to perform a secondary function while th Q22-bus I/O map is initialized, namely to synchronize an auxiliar processor with its bootstrap host. See section 3.5.1.7 for additiona information.

3.5.1 Primary Bootstrap Program (VMB) -

VMB is the KA630 primary bootstrap. It is executed as the first part c a two part system bootstrap operation. VMB contains the code that:

1. initializes the system control block (SCB),

2. initializes an extended restart parameter block (RPB),

3. initializes a PFN (page frame number) bit map and the relevan extended RPB fields,

4. selects a bootstrap device, and

5. performs a Files-11 ODS2, boot block, ROM or down-line load c the secondary bootstrap.

The secondary bootstrap continues the bootstrap operation. For KA63 systems, primary bootstrap operations are defined by VMB, secondar bootstrap are defined by the operating system being booted.

VMB finds the bootstrap device in one of three ways:

1. If the bootstrap is the result of a console bootstrap comman and a device name is specified in the command, that device i searched for the secondary bootstrap.

2. If not the result of a console bootstrap command or if n device name is specified, VMB searches first for a bootable disk, then for a TK50 (Maya) tape unit, then for MRV11 PROM, and finally for a DEQNA for a down-line bootstrap.

3. If the bootstrap is the result of a halt with CPMBX<1:0> equa to 2 - a request from the operating system to reboot the syste - the bootstrap device used to previously bootstrap the operating system is used (as well as the same command flags).

When a VMB bootstrap attempt fails, it halts.

3.5.1.1 Bootstrap Devices -

The following bootstrap devices are supported:

1. RQDX, QDA and Aztec MSCP disk controllers. VMB can boot fr any disk unit supported by a MSCP QD or Aztec disk controlle Units supported by RQDX are RX50, RD51, RD52 and RD53. Uni supported by QDA are tbs. For purposes of the BOOTSTR command, units are designated DUA0, DUA1, and so on. The firs controller must be configured at the Q22-bus address of 17721 (octal) and Q22-bus interrupt vector of 154 (octal) Additional controllers are located in floating CSR and vect space.

2. DEQNA Ethernet adapter. This controller connects to a Ethernet cable. For purposes of the BOOTSTRAP command, thi device is designated XQA0. The controller must be configure at Q22-bus address 1774440 (octal) and Q22-bus vector of tbs.

3. MRV11 PROM. This is a Programmable Read Only Memory boar For purposes of the BOOTSTRAP command it is designated PRA0.

4. TMSCP tape controller (TK50). VMB can boot from any tape uni supported by the the TK50 TMSCP tape controller. For purpose of the BOOTSTRAP command it is designated tbs. The controlle must be configured at Q22-bus address tbs and Q22-bus interru vector tbs. Additional controllers are not be supported.

3.5.1.2 Bootstrap Command Fl - When a bootstrap is invoked via BOOTSTRAP command, the user can specify several boot command flags b bit encoding the flags in a flag word specified with the "/R5: qualifier. These command flags are described in table 4.

Table 4: VMB Bootstrap Command Flags

Bit Number	Value (Hex)	Flag Name	Description
0	00000001	CONVERSATION	Conversational bootstrap.

Table 4 (cont.)

Bit Number	Value (Hex)	Flag Name	Description
3	00000008	BOOTBLOCK	Secondary bootstrap from boo block. When this bit is set, VM reads logical block number 0 of th boot device and tests it fo conformance with the boot bloc format. If in conformance, th block is executed to continue th bootstrap. No attempt to perform Files-11 bootstrap is made.
4	00000010	DIAGNOSTIC	Diagnostic bootstrap. When this bi is set the secondary bootstrap i called [SYS0.SYSMAINT]DIAGBOOT.EXE
6	00000040	HEADER	Image header. If this bit is no set, VMB transfers control to th first location of the secondar bootstrap. If this bit is set, i transfers control to the addres specified by the file's imag header.
8	00000100	SOLICIT	File name solicit. When this bit i set, VMB prompts the operator fo the name of the secondary bootstra file.
9	00000200	HALT	Halt before transfer. When this bi is set, VMB halts befor transfering control to th secondary bootstrap.
<31:28>	X0000000	TOPSYS	X can be any value from 0 through (hex). This flag changes the to level directory name for syste disks with multiple operatin systems. For example, if X=1, th top level directory name is [SYS1 ...].

3.5.1.3 Booting From Disk -

For VMB to boot using a MSCP disk controller, the first controller mus be configured at 772150(octal) and subsequent controllers must b properly configured in their appropriate floating CSR and vector slots

Table 4 (cont.)

Bit Number	Value (Hex)	Flag Name	Description
3	00000008	BOOTBLOCK	Secondary bootstrap from bo block. When this bit is set, V reads logical block number 0 of t boot device and tests it fo conformance with the boot blo format. If in conformance, t block is executed to continue t bootstrap. No attempt to perform Files-11 bootstrap is made.
4	00000010	DIAGNOSTIC	Diagnostic bootstrap. When this b is set the secondary bootstrap called [SYS0.SYSMAINT]DIAGBOOT.EX
6	00000040	HEADER	Image header. If this bit is n set, VMB transfers control to t first location of the secondar bootstrap. If this bit is set, transfers control to the addres specified by the file's ima header.
8	00000100	SOLICIT	File name solicit. When this bit set, VMB prompts the operator fo the name of the secondary bootstra file.
9	00000200	HALT	Halt before transfer. When this b is set, VMB halts befor transfering control to t secondary bootstrap.
<31:28>	X0000000	TOPSYS	X can be any value from 0 through (hex). This flag changes the to level directory name for syste disks with multiple operatin systems. For example, if X=1, t top level directory name is [SYS ...].

3.5.1.3 Booting From Disk -

For VMB to boot using a MSCP disk controller, the first controller mus be configured at 772150(octal) and subsequent controllers must b properly configured in their appropriate floating CSR and vector slots

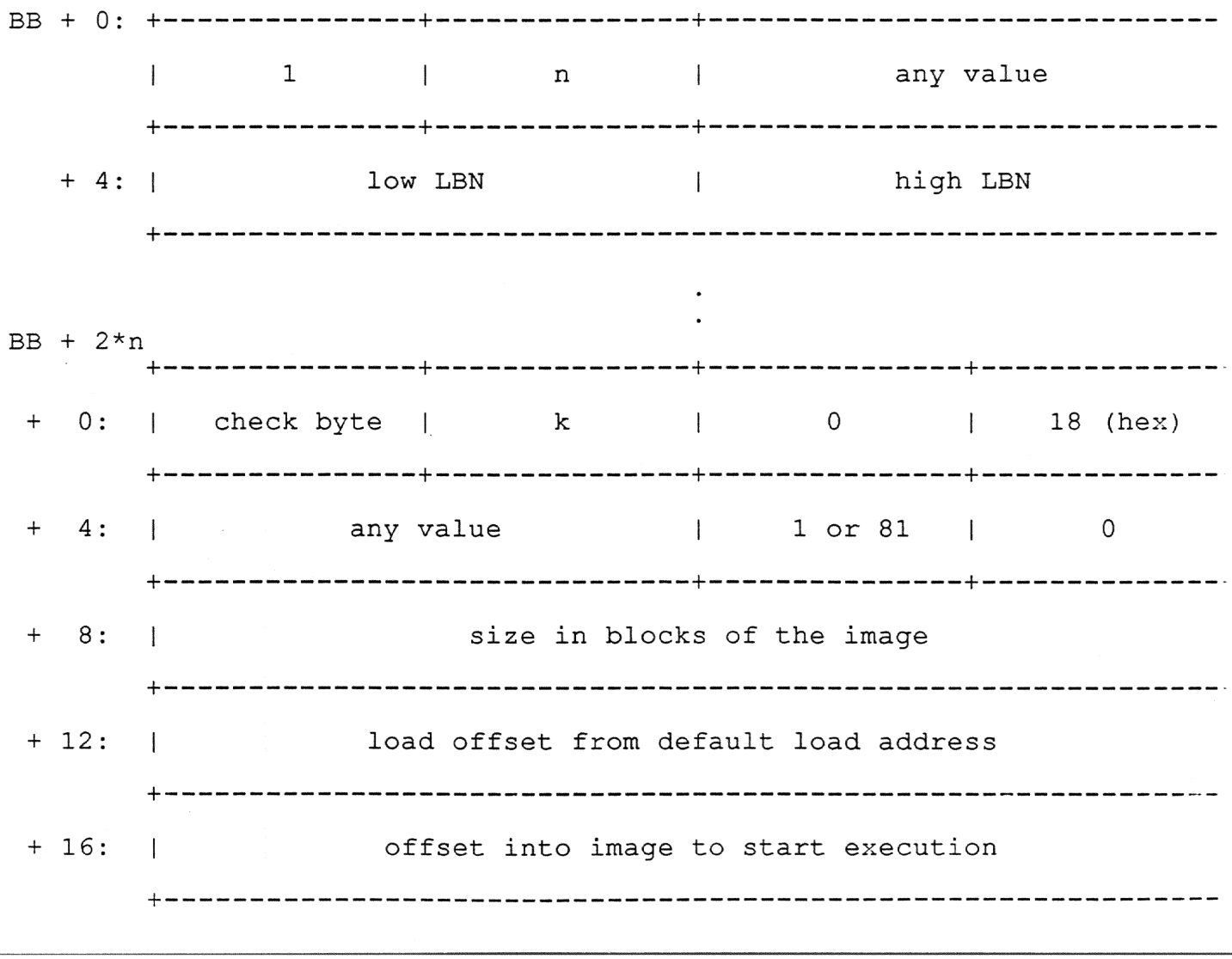
When VMB determines that a controller is present, it searchs in order c
increasing unit number for an accessible unit with a removable volume
If it finds such a unit with a volume present, VMB proceeds as describe
below. If it finds no such volume, it repeats the search but this tim
it checks for non-removable volumes. If by this time no accessibl
volume is found, it checks for the next controller and repeats th
checks described. If no more controllers are found, the disk boo
fails.

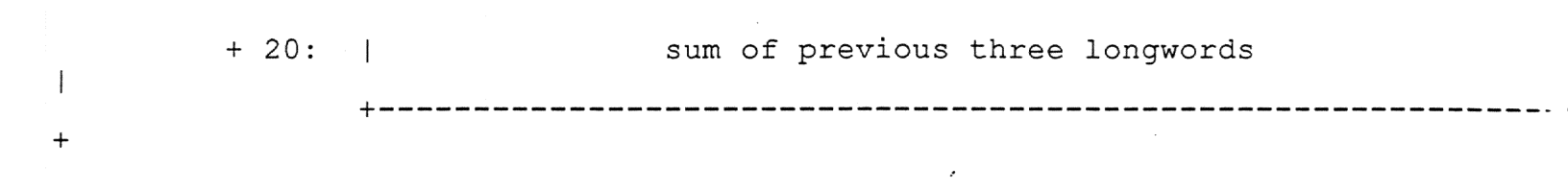
If an accessible volume is located, VMB then determines if it is
Files-11 volume. If it is, it searches the volume for "[SYS0.SYSEXE
SYSBOOT.EXE" which contains the secondary bootstrap. If this file i
found, VMB loads and executes it (performs a secondary bootstrap).

If the volume is not a Files-11 volume, VMB then checks logical bloc
number zero of the volume for a valid bootblock (see figure 5). If th
bootblock is a valid bootblock VMB loads and executes the secondar
bootstrap specified in the bootblock. If there is not a valid bootbloc
present, the search resumes for the next accessible disk.

Note that the bootstrap process can be altered by boot command flags
Information on the effect of boot command flags is given in table 4.

Figure 5: Bootblock Format





BB+0: These two bytes can have any value.

BB+2: This value is the word offset from the start

the bootblock to the identification are described below.

BB+3: This byte must be one.

BB+4: This longword contains the logical block numbe (word swapped) of the secondary image.

BB+(2*n)+0: This byte defines the expected instruction set 18(hex) means VAX.

BB+(2*n)+1: This byte defines the expected controller type 0 means unknown.

BB+(2*n)+2: This byte defines the file structure on th volume, it may be any value.

BB+(2*n)+3: This byte must be the ones complement of the su of the previous three bytes.

BB+(2*n)+4: This byte must be zero.

BB+(2*n)+5: This byte must be 1 or 81 (hex). This byt defines the version number of the forma standard and the type of disk. The version i 1, the high bit is 0 for single sided, 1 fo double sided.

BB+(2*n)+6: These two bytes must be zero.

BB+(2*n)+8: This entry is a longword containing the size i blocks of the secondary bootstrap image.

BB+(2*n)+12: This entry is a longword containing a loa offset (usually zero) from the default loa address of the secondary bootstrap.

BB+(2*n)+16: This entry is a longword containing the byt offset into the secondary bootstrap wher execution is to begin.

BB+(2*n)+20: This entry is a longword containing the sum o the previous three longwords.

3.5.1.4 Booting From Tape -

If no bootable disk is found, VMB attempts to bootstrap from a TK5 tape. The TK50 must be configured at <tbs> octal to be recognized b VMB.

If a TK50 is present, VMB determines if a tape is loaded and the unit i online. If so, VMB rewinds the tape and searches for the fil TAPEBOOT.EXE. (The user may specify an alternative file name by settin the SOLICIT bit in the software command register [see table 4]). I this file is found, VMB loads and executes it. Normally this file woul contain a program to load an operating system from tape onto a syste disk.

Note that a TAPEBOOT.EXE program to load a disk from tape is not a par of this project.

If a user has both disks and tape, and a disk is bootable, to boot fro tape the user must either take all bootable disks offline or explicitl boot the TK50 via the console boot command.

3.5.1.5 Booting From PROM -

If neither disk nor tape is bootable, VMB checks for a PROM bootstrap.

To locate a PROM bootstrap, VMB searches the Q22-bus address range fro low to high addresses in 16 page chuncks looking for readable memory If the first six longwords of any such page contains a valid "signatur block" (identical to the second part of the bootblock format, see figur 5), VMB copies the PROM image to main memory and then transfers contro to it.

Note that while defined as a MRV11 PROM or equivalent bootstrap, VM doesn't actually require that the signature block or the bootstrap cod be in PROM, it could be in ROM, nonvolatile RAM or, as described i section 3.5.1.7, it could be loaded into another KA630's RAM and mappe to the Q22-bus.

3.5.1.6 Booting From DEQNA -

If no other bootstrap device is found, VMB attempts to bootstrap fro the DEQNA ethernet controller. In this case, the secondary bootstrap i downline loaded from a host on the Ethernet using DECNET Low-level Maintenance Protocol (MOP) Version 3.0. The DEQNA module must b configured at Q22-bus address 774440 (octal).

The downline load process consists of the following steps:

1. VMB performs local testing of the DEQNA. If the tests fail the bootstrap attempt fails and the three LED's on the DEQN are set according to the problem detected. The LED setting and their interpretations are:

   - 3 LED's on: DEQNA initialization failure.

- o 2 LED's on: internal loopback failure.
- o 1 LED on: external loopback failure.

2. VMB transmits a Program Request MOP message over the Ethernet The message destination is the Load Assistant Multicast addres AB-00-00-01-00-00. The message source address is the DEQNA' station address (from DEQNA PROM). The MOP program type i operating system.

3. VMB waits approximately 30 seconds to receive a response. I it does not receive a response it retransmits the request ever thirty seconds for a total of 2 minutes. If a response is no received in two minutes, the bootstrap fails.

4. VMB accepts MOP Load Messages and loads the data into memor terminating when the final message is received as indicated i the MOP message protocol. If the interval between loa messages exceeds 30 seconds, VMB restarts the DEQNA bootstra at step 2.

3.5.1.7 Booting An Auxiliary Processor -

VMB bootstraps an auxiliary processor via the ROM bootstrap protocol Recall from section 3.5 the Q22-bus initialization algorithm:

1. Turn on IPCR<8>, the halt flag.
2. Disable the I/O map by clearing IPCR<5>.
3. Initialize the I/O map.
4. Enable the I/O map by setting IPCR<5>.
5. If an auxiliary processor, loop until the IPCR<8> bit i cleared.

(Note that whenever the console program is entered, it turns of IPCR<8>.) Steps 1 and 5 ensure that an auxiliary will loop until som other processor clears IPCR<8>. When another processor, the bootstra host, clears IPCR<8>, the auxiliary proceeds with the bootstrap. Thi synchronization gives the arbiter processor control over th bootstrapping of all auxiliary processors.

An auxiliary processor cannot directly bootstrap itself from any of th normal bootstrap devices so VMB on an auxiliary checks only for the RO bootstrap described in section previously. The ROM bootstrap may b either a block of nonvolatile memory on the Q22-bus bus or the bootstra host can construct an equivalent bootstrap in RAM. In either case th

auxiliary will not proceed with the bootstrap until the bootstrap hos clears IPCR<8>. The bootstrap host in turn should not clear th auxiliary's IPCR<8> unless IPCR<5> is clear.

3.5.2 Secondary Bootstrap Program -

The secondary bootstrap program is invoked as the second part of system bootstrap. Following successful execution of the primar bootstrap, the secondary bootstrap has either been loaded into memory o located in ROM. It is the responsibility of the secondary bootstrap t complete the bootstrap of the processor.

VMB calls the secondary bootstrap with the processor in the followin state:

- The processor is running in kernel mode on the interrupt stac at IPL 31(hex).
- R11 contains the base address of the extended RPB (figure 6 built by VMB.
- AP contains the address of the secondary bootstrap argumen list (figure 7).
- SP contains the address of the top of the stack plus 4, whic is also the address of the beginning of the secondary bootstra (figure 8 for a map of memory).
- SCBB (internal processor register) contains the address of th SCB built by VMB.

Note that the first four longwords of the VMB built extended RPB woul not be recognized as a valid RPB by the console restart algorithm. I is up to the secondary bootstrap or the operating system itself t complete the RPB if automatic restart is desired.

Figure 6: Extended RPB

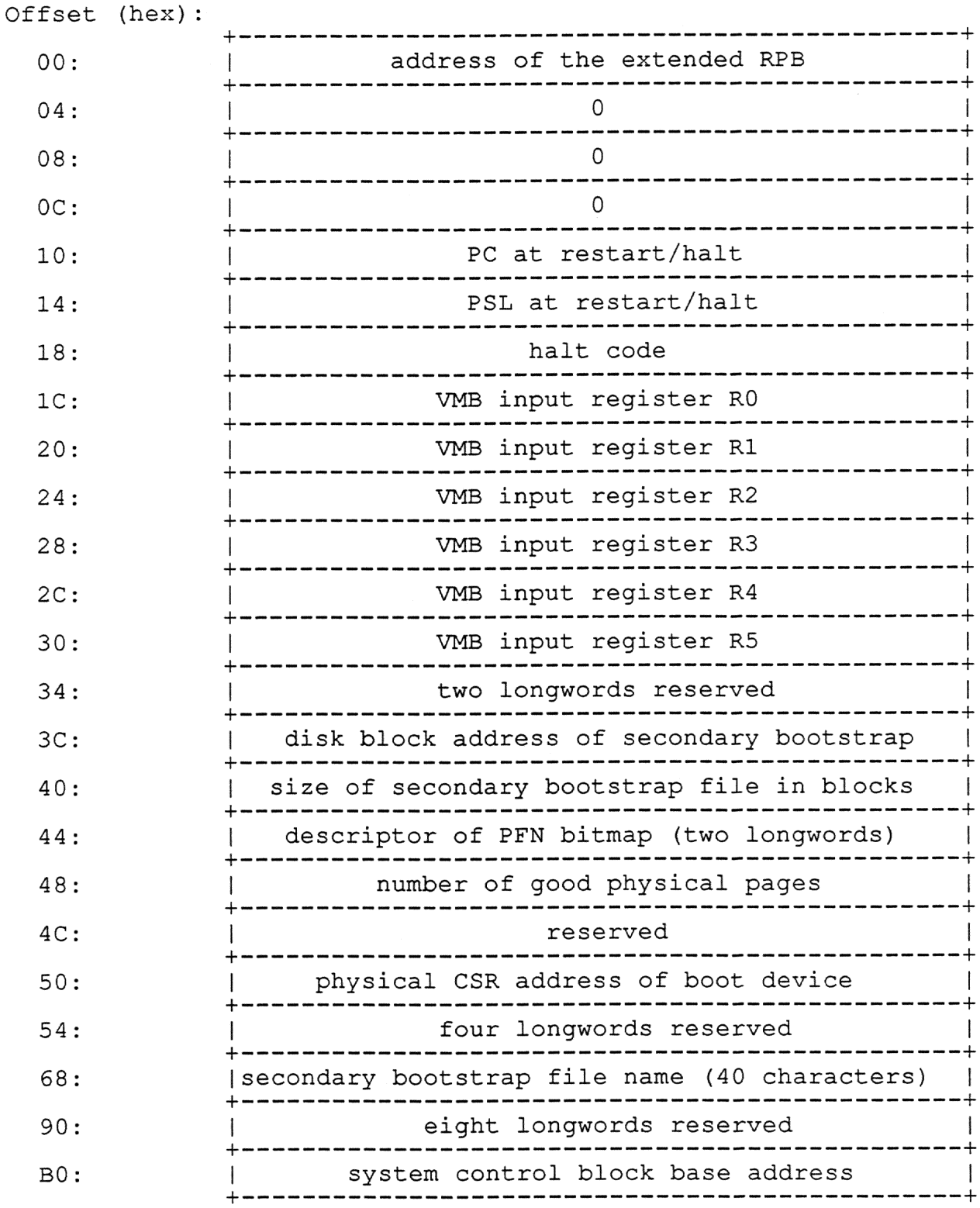


Figure 7: Secondary Bootstrap Argument list

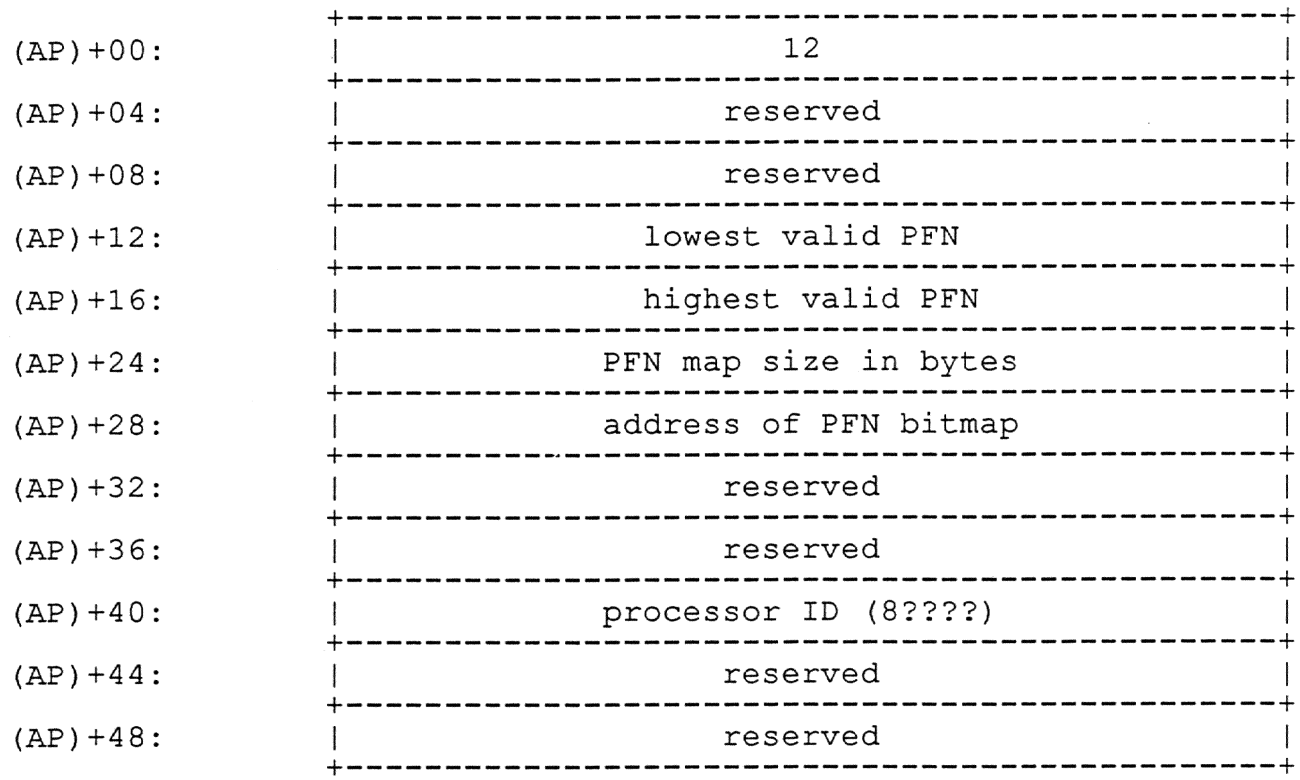
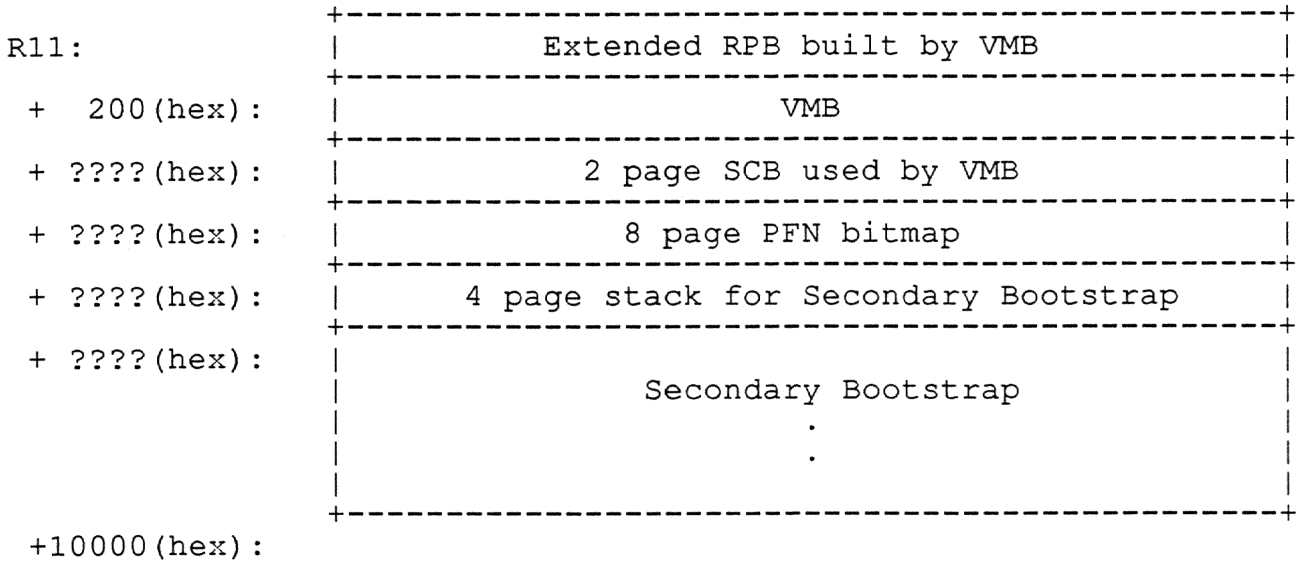


Figure 8: Secondary Bootstrap Memory Map



3.6 Console I/O Mode (System Halted)

When the KA630 is "halted", the operator controls the system through th console terminal using the console command language. The consol terminal is in "Console I/O Mode". The console prompts the operator fo input with the string ">>>".

3.6.1 Console Control Characters -

In console I/O mode, several characters have special meanings.

- carriage return - ends a command line. No action is taken on command until after it is terminated by a carriage return. null line terminated by a carriage return is treated as valid, null command. No action is taken, and the consol re-prompts for input. Carriage return is echoed as carriag return, line feed.

- rubout - when the operator types rubout, the console delete the character that the operator previously typed. What appear on the console terminal depends on whether the terminal is video terminal or a hardcopy terminal.

  For hard copy terminals, when a rubout is typed, the consol echoes with a backslash (\), followed by the character bein deleted. If the operator types additional rubouts, th additional characters deleted are echoed. When the operato types a non-rubout character, the console echoes anothe backslash, followed by the character typed. The result is t echo the characters deleted, surrounding them with backslashes For example:

  The operator types: EXAMI;E<rubout><rubout>NE<CR>

  The console echoes: EXAMI;E\E;\NE<CR>

  The console sees the command line: EXAMINE<CR>

  For video terminals, when rubout is typed the previou character is erased from the screen and the cursor is restore to its previous position.

  The console does not delete characters past the beginning of command line. If the operator types more rubouts than ther are characters on the line, the extra rubouts are ignored. I a rubout is typed on a blank line, it is ignored.

- control-U - the console echoes "^U<CR>", and deletes the entir line. If control-U is typed on an empty line, it is echoed and otherwise ignored. The console prompts for anothe

command.

- control-S - stops output to the console terminal unt control-Q is typed. Control-S and control-Q are not echoe Control-C, control-O, and BREAK also clear control-S.

- control-Q - resumes output to the console terminal. Additiona control-Q's are ignored. Control-S and control-Q are n echoed.

- control-O - causes the console to throw away transmissions t the console terminal until the next control-O is entere Control-O is echoed as "^O"<CR> when it disables output, but i not echoed when it reenables output. Output is reenabled i the console prints an error message, or if it prompts for command from the terminal. Displaying a REPEAT command do not reenable output. When output is reenabled for reading command, the console prompt is displayed. Output is als enabled by entering program I/O mode, by BREAK and b control-C. Control-O clears control-S.

- control-R - causes the console to echo <CR><LF> followed by th current command line. This function can be used to improve th readability of a command line that has been heavily edited.

- control-C - causes the console to echo "^C" and to abo processing of a command. Control-C has no effect as part of binary load data stream. Control-C clears control-S, an reenables output stopped by control-O. When control-C is type as part of a command line, the console deletes the line as i does with control-U.

- BREAK - If the console is in console I/O mode, BREAK i equivalent to control-C, but is echoed as "^P". If the consol is in program I/O mode and halt is disabled, BREAK is ignore If the console is in program I/O mode and halt is not disable BREAK causes the processor to halt and enter console I/O mode.

Control characters are typed by pressing the character key whil simultaneously holding down the control key.

If an unrecognized control character is typed (a control character her means a character with an ASCII code less than 32 decimal [C0] c between 128 and 159 decimal [C1]) it is echoed as up arrow followed b the character with ASCII code 64 greater. For example, BEL (ASCII cod 7) is echoed as "^G", since capital G is ASCII code 7+64=71. When control character is deleted with rubout, it is echoed the same way After echoing the control character, the console processes it like normal character. Unless the control character is part of a comment the command will be invalid, and the console will respond with an erro message. Note that control codes from 128 to 159, the C1 control codes cannot be entered by any present Digital terminal. The fact that bot

the character with code 7 and the character with code 135 will both ec as "^G" is not expected to have any practical consequences.

3.6.2 Console Command Syntax -

The console accepts commands of lengths up to 80 characters. Longe commands are responded to with an error message. The count does n include rubouts, rubbed out characters, or the terminating carria return.

Commands may be abbreviated. Abbreviations are formed by droppi characters from the end of a keyword. All commands are recognized fr their first character.

Multiple adjacent spaces and tabs are treated as a single space by t console. Leading and trailing spaces and tabs are ignored.

Command qualifiers can appear after the command keyword, or after a symbol or number in the command.

All numbers (addresses, data, counts) are in hexadecimal. (Note though, that symbolic register names include decimal digits.) Hex digi are 0 through 9, and A through F. The console does not distinguis between upper and lower case either in numbers or in commands. Both a accepted.

3.6.3 Console Command Keywords -

Processor control commands:

- INITIALIZE
- START <address>
- CONTINUE
- HALT
- BOOT <device>
- UNJAM

Data transfer commands:

- EXAMINE <address>
- DEPOSIT <address> <data>

- X <address> <count>

Console control commands:

- FIND
- REPEAT <command>
- TEST
- ! <comment>

3.6.4 References To Processor Registers And Memory -

The KA630 console is implemented by macrocode executing from ROM. Fo this reason, the actual processor registers may not be modified by th command interpreter. When the console is entered, the console saves th processor registers in console memory and all command references to the are directed to the corresponding scratch page locations, not to th registers themselves. When the console reenters program mode, the save registers are restored and any changes become operative only then References to processor memory are handled normally except where note below.

The console uses memory at the top of the available system memory fo its own use (see figure 1). References to the console memory pages b EXAMINE and DEPOSIT commands must be qualified by the "/U" qualifier (Access is primarily to simplify debugging of the console program.) Th binary load and unload command may not reference the console memor pages.

3.6.5 Console Commands -

3.6.5.1 BOOT -

BOOT [<qualifier list>] [<device>]

The device specification is of the format 'ddcu', where 'dd' is a tw letter device mnemonic, 'c' is an optional one digit controller number and 'u' is a one digit unit number.

The console initializes the processor and starts VMB running. (See th section on System Bootstrapping.) VMB boots the operating system fro the specified device. The default bootstrap device is determined a described in the section on system bootstrapping.

Qualifiers:

- /R5:<data> - After initializing the processor and befor starting VMB, R5 is loaded with the specified data. Thi allows a console user to pass a parameter to VMB. (To remai compatible with previous processors, /<data> will also b recognized to have the same result.)

3.6.5.2 CONTINUE -

CONTINUE

The processor begins instruction execution at the address currentl contained in the program counter. Processor initialization is no performed. The console enters program I/O mode.

If upon entry the console program is unable to preserve the machin state, either because it was a power-up or because it was unable t locate a scratch page on the interrupt stack, the CONTINUE command wil be rejected.

3.6.5.3 DEPOSIT -

DEPOSIT [<qualifier list>] <address> <data>

Deposits the data into the address specified. If no address space o data size qualifiers are specified, the defaults are the last addres space and data size used in a DEPOSIT or EXAMINE command. Afte processor initialization, the default address space is physical memory the default data size is long, and the default address is zero.

If the specified data is too large to fit in the data size to b deposited, the console ignores the command and issues an error response If the specified data is smaller that the data size to be deposited, it is extended on the left with zeros.

The address may also be one of the following symbolic addresses:

- PSL - the processor status longword. No address spac qualifier is legal. When PSL is examined, the address space i identified as "M" (machine dependent).

- PC - the program counter (general register 15). The addres space is set to /G.

- SP - the stack pointer (general register 14). The addres space is /G.

- Rn - general register 'n'. The register number is in decimal The address space is /G. For example:

  D R5 1234 is equivalent to D/G 5 1234

  D R10 6FF00 is equivalent to D/G A 6FF00

- '+' - the location immediately following the last locatio referenced in an examine or deposit. For references t physical or virtual memory spaces, the location referenced i the last address, plus the size of the last reference (1 fo byte, 2 for word, 4 for long). For other address spaces, th address is the last address referenced, plus one.

- '-' - the location immediately preceding the last locatio referenced in an examine or deposit. For references t physical or virtual memory spaces, the location referenced i the last address minus the size of this reference (1 for byte 2 for word, 4 for long). For other address spaces, the addres is the last addressed referenced minus one.

- '*' - the location last referenced in an examine or deposit.

- '@' - the location addressed by the last location referenced i an examine or deposit.

Qualifiers:

- /B - The data size is byte.

- /W - The data size is word.

- /L - The data size is longword.

- /V - The address space is virtual memory. All access an protection checking occur. If the access would not be allowe to a program running with the current PSL, the console issue an error message. Virtual space DEPOSITs cause the PTE<M> bi to be set. If memory mapping is not enabled, virtual addresse are equal to physical addresses.

- /P - The address space is physical memory.

- /I - The address space is internal processor registers. Thes are the registers addressed by the MTPR and MFPR instructions.

- /G - The address space is the general register set, R0 throug PC.

- /U - Access to the console scratch page is allowed. Thi qualifier also disables virtual address protection checks.

- /N:<count> - The address is the first of a range. The consol deposits to the first address, then to the specified number o succeeding addresses. Even if the address is the symboli address "-", the succeeding addresses are at larger addresses The symbolic address specifies only the starting address, no the direction of succession. For repeated references t preceding addresses, use "REPEAT DEPOSIT - <data>".

  For example:

D/P/B/N:1FF 0 0	Clears the first 512 bytes of physica memory.
D/V/L/N:3 1234 5	Deposits 5 into 4 longwords starting a virtual address 1234.
D/N:8 R0 FFFFFFFF	Loads general registers R0 through R with -1.
D/N:200 - 0	Starting at previous address, clear 51 bytes.

If conflicting address space or data sizes are specified, the consol ignores the command and issues an error response.

3.6.5.4 EXAMINE -

EXAMINE [<qualifier list>] [<address>]

Examines the contents of the specified address. If no address i specified, "+" is assumed. The address may also be one of the symboli addresses described under DEPOSIT.

Qualifiers:

The same qualifiers may be used on EXAMINE as may be used on DEPOSIT.

RESPONSE: <tab><address space identifier> <address> <tab> <data>

The address space identifier can be:

- P - physical memory. Note that when virtual memory i examined, the address space and address in the response are th translated physical address.
- G - general register.
- I - internal processor register.
- M - machine dependent (used only for display of the PSL).

3.6.5.5 FIND -

FIND [<qualifier list>]

The console searches main memory starting at address zero for page-aligned 64 kilobyte segment of good memory, or a restart parameter block (RPB). If the segment or block is found, its address plus 512 is left in SP. If the segment or block is not found, an error message is issued, and the contents of SP are UNPREDICTABLE. If no qualifier is specified, /RPB is assumed.

Qualifiers:

- /MEMORY - search memory for a page aligned segment of good memory, 64 kilobytes in length. The search includes a read/write test of memory and leaves the contents of memory UNPREDICTABLE.
- /RPB - search memory for a restart parameter block. See the section 3.4 for the search algorithm. The search leaves the contents of memory unchanged.

3.6.5.6 INITIALIZE -

INITIALIZE

A processor initialization is performed. The following registers are set (all values are hexadecimal):

PSL	041F0000
IPL	1F
ASTLVL	4
SISR	0
ICCS	0
RXCS	0
TXCS	80
MAPEN	0

All other registers are UNPREDICTABLE.

The previous console reference defaults (the defaults used to fill in unsupplied qualifiers for DEPOSIT and EXAMINE commands) are set to physical address, longword size and address 0.

3.6.5.7 HALT -

HALT

The HALT command has no affect, the processor is already halted when i
console I/O mode.

RESPONSE: Already halted

3.6.5.8 REPEAT -

REPEAT <command>

The console repeatedly displays and executes the specified command. Th
repeating is stopped by the operator typing control-C. Any vali
console command may be specified for the command with the exception o
the REPEAT command.

RESPONSE: <dependent upon command specified>

3.6.5.9 START -

START [<address>]

The console starts instruction execution at the specified address. I
no address is given, the current PC is used. If no qualifier i
present, macroinstruction execution is started. If memory mapping i
enabled, macroinstructions are executed from virtual memory. The STAR
command is equivalent to a DEPOSIT to PC, followed by a CONTINUE. N
INITIALIZE is performed.

3.6.5.10 TEST -

TEST [<test number>]

The console invokes a diagnostic test program denoted by <test number>
Valid test numbers are 3 through 7 and hexidecmal B.

See "KA630 ROM Diagnostic Specification" for information on diagnosti
tests and their corresponding test number codes.

3.6.5.11 UNJAM -

An I/O bus reset is performed.

3.6.5.12 Binary Load And Unload Command -

X <address> <count> <CR> <checksum>

The X command is for use by automatic systems communicating with th console. It is not intended for use by operators. The console loads o unloads (that is, writes to memory, or reads from memory) the specifie number of data bytes, starting at the specified address.

If bit 31 of the count is clear, data is to be received by the consol and deposited into memory. If bit 31 of the count is set, data is to b read from memory and sent by the console. The remaining bits in th count are a positive number indicating the number of bytes to load o unload.

The console accepts the command upon receiving the carriage return. Th next byte the console receives is the command checksum, which is no echoed. The command checksum is verified by adding all comman characters, including the checksum, (but not including the terminatin carriage return or rubouts or characters deleted by rubout), into an bit register initially set to zero. If no errors occur, the result i zero. If the command checksum is correct, the console responds with th input prompt and either sends data to the requester or prepares t receive data. If the command checksum is in error, the console respond with an error message. The intent is to prevent inadvertent operato entry into a mode where the console is accepting characters from th keyboard as data, with no escape sequence possible.

If the command is a load (bit 31 of the count is clear), the consol responds with the input prompt, then accepts the specified number o bytes of data for depositing to memory, and an additional byte o received data checksum. The data is verified by adding all dat characters and the checksum character into an 8 bit register initiall set to zero. If the final contents of the register is non-zero, th data or checksum are in error, and the console responds with an erro message.

If the command is a binary unload (bit 31 of the count is set), th console responds with the input prompt, followed by the specified numbe of bytes of binary data. As each byte is sent it is added to a checksu register initially set to zero. At the end of the transmission, the 2' complement of the low byte of the register is sent.

If the data checksum is incorrect on a load, or if memory errors or lin errors occur during the transmission of data, the entire transmission i completed, and then the console issues an error message. If an erro occurs during loading, the contents of the memory being loaded ar UNPREDICTABLE.

Echo is suppressed during the receiving of the data string an checksums.

It is possible to control the console through the use of the consc control characters (control-C, control-S, control-O, etc.) during binary unload. It is not possible during a binary load, as all receiv characters are valid binary data.

Data being loaded with a binary load command must be received by t console at a rate of at least one byte per second. The command checks that precedes the data must be received by the console within 10 secon of the <CR> that terminates the command line. The data checksum must b received within 10 seconds of the last data byte. If any of thes timing requirements are not met the console aborts the transmission b issuing an error message and prompting for input.

The entire command, including the checksum, may be sent to the consc as a single burst of characters at the console's specified characte rate. The console is able to receive at least 4K bytes of data in single 'X' command.

3.6.5.13 Comment -

! <comment>

The comment command is ignored. It is used to annotate console I, command sequences.

3.6.6 Console Errors And Error Messages -

The console can issue error messages in response to commands. Thes error messages are listed in table 5.

Table 5: Console Error Messages

Code	Message	Description
?15	CORRPTN	The console program database has been corrupted, the console program does a powerup initialization to rebuild its database.
?16	ILL REF	The requested reference would violate virtual memory protection, the address is not mapped, the reference is invalid in the specified address space, or the value is invalid in the specified destination.
?17	ILL CMD	The command string cannot be parsed.
?18	INV DGT	A number has an invalid digit.

Table 5 (cont.)

Code	Message	Description
?19	LTL	The command was too large for the console to buffer. The message is issued only after receipt of the terminating carriage return.
?1A	ILL ADR	The address specified falls outside the limits of the address space.
?1B	VAL TOO LRG	The value specified does not fit in the destination.
?1C	SW CONF	For example, two different data sizes are specified with an EXAMINE command.
?1D	UNK SW	The switch is unrecognized.
?1E	UNK SYM	The symbolic address in an EXAMINE or DEPOSIT is unrecognized.
?1F	CHKSM	The command or data checksum of an X command is incorrect. If the data checksum is incorrect, this message is issued, and is not abbreviated to "Illegal command".
?20	HLTED	The operator entered a HALT command.
?21	FND ERR	A FIND command failed either to find the RPB or 64 kb of good memory.
?22	TMOUT	During an X command, data failed to arrive in the time expected.
?23	MEM ERR	Parity error detected.

Some console commands may result in errors. For example, if a memory error occurs as the result of a console command, the console will respond with an error message.

3.6.7 Halts And Halt Messages -

Whenever the processor halts, the console prints the response "PC = <PC>". For example:

```
?06 HLT INST
    PC = 800050D3
```

The number preceding the halt message is the halt code, and is passed t the operating system on a restart. The halt messages are shown in tabl 6.

Table 6: KA630 Halt Messages

Code	Message	Description
?02	EXT HLT	BREAK was typed on the console, QBINIT or QBHALT was asserted.
?04	ISP ERR	In attempting to push state onto the interrupt stack during an interrupt or exception, the processor discovered that the interrupt stack was mapped NO ACCESS or NOT VALID.
?05	DBL ERR	The processor attempted to report a machine check to the operating system, and a second machine check occurred.
?06	HLT INST	The processor executed a HALT instruction in kernel mode.
?07	SCB ERR3	The vector had bits <1:0> equal to 3.
?08	SCB ERR2	The vector had bits <1:0> equal to 2.
?0A	CHM FR ISTK	A change mode instruction was executed when PSL<IS> was set.
?0B	CHM TO ISTK	The exception vector for a change mode had bit <0> set.
?0C	SCB RD ERR	A hard memory error occurred while the processor was trying to read an exception or interrupt vector.
?10	MCHK AV	An access violation or an invalid translation occured during machine check exception processing.
?11	KSP AV	An access violation or an invalid translation occured during processing of of an invalid kernel stack pointer exception.

3.7 Program I/O Mode (System Running)

When the processor is not executing instructions from the consol program ROM, it is in "program I/O mode" in which all termina

interaction is handled by the operating system. In program I/O mode the console terminal behaves like any other operating system terminal. If halts are disabled, BREAK is ignored. If halts are enabled, BREAK causes the processor to "halt", that is the KA630 enters console I/O mode.

4 PUBLICATIONS

tbd

5 PACKAGING

NOT APPLICABLE - This product resides in ROM and is shipped as part of the processor board.

6 INSTALLABILITY

NOT APPLICABLE - This product resides in ROM and is shipped as part of the processor board.

7 EASE OF USE

The users of the KA630 console program include many who are sophisticated in their use and understanding of computers. These users are Engineering, Manufacturing and Field Service personnel as well as many customers. These users are not intimidated by the console command language and its messages nor by the messages generated by the console diagnostics. The relative complexity of these components of the console system are perceived as worth the diagnostic power that they provide. It is for these users that these features are provided. Indeed these features have evolved over time to satisfy these users and therefore for this user group, the KA630 console program is regarded as "easy to use".

Another class of users however are those users who know little about computers and who may be intimidated by them. These users would like to simply flip a switch and know that the computer will turn on and they can proceed with their work. These users are not expected to use the console command language, nor are they expected to understand any diagnostic messages.

If a KA630 is to be used by a user from the latter group, halts must be disabled, the appropriate bootstrap device configured, and the proper baud rate must be set. Establishing this configuration for the unsophisticated user may be done in manufacturing (the preferred

approach), by Field Service personnel, by a sophisticated user, or by a unsophisticated user following a well designed and documente installation procedure.

Once properly configured, assuming no hardware problems are detected b the diagnostics on power up, figure 9 shows what the user will se displayed.

Figure 9: Console Display on Successful Power-Up

```
KA630.xx

Performing normal system tests.

  7..6..5..4..3..

Tests completed.

Loading system software.

  2..1..0

<Operating system specific output>
```

The first line identifies the processor and the version number (xx) c the console program ROM. The next line explains that the system i performing NORMAL tests. The count-down sequence reassures the use that the system is progressing through its tests, that the system hasn' "gone away", and documents which tests are executed. When diagnostic are completed, the console notifies the user that the diagnostic sucessfully completed. The "Loading system software." message indicate the beginning of the bootstrap sequence. The execution of the bootstra sequence causes the remaining digits of the countdown to be displayed Because successful completion of a bootstrap occurs in the context c the operating system bootstrapped, a confirming message indicating tha the system startup has completed can only be issued by the bootstrappe operating system.

Figure 10 shows the display generated if any hard errors are detecte during any test (test 7 for example).

Figure 10: Console Display on Power-Up with Hard Errors

```
KA630.xx

Performing normal system tests.

  7..

?<subtest> <p1> <p2> <p3>

Failure.
Normal operation not possible.
```

In the case of fatal problems detected by the diagnostics, the countdow sequence is interrupted and a diagnostic message is displayed. Th diagnostic message is composed of a question mark, a subtest code numbe and up to three parameters for use by diagnostic personel. More tha one such error message is possible but unlikely. The summary messag which follows indicates that the test failed and that normal operatio is not possible. The console program then hangs.

8 PERFORMANCE

All diagnostic checks will be performed in less than tbd. Diagnosti coverage of the processor itself will be tbd percent, coverage of th memory system will be tbd percent, and coverage of the Q22-bus map wil be tbd.

The console responds to all commands within 1 second.

When the console power-up display is displayed, no more than 5 second shall pass without some output to the screen (additional periods may b displayed during the countdown if necessary).

9 RELIABILITY

Errors detected by this program are classified as user errors, har errors, or catastrophic errors. User errors are errors arising fro invalid input from the user. All user errors are diagnosed and the use is notified by a diagnostic message on the console terminal. All suc errors are recoverable.

Hard errors are errors detected by the program which whil unrecoverable, do not prevent the continuation of the program. A example of a hard error is the detection of an unexpected non-existen memory error when in console I/O mode. The console program notifies th user of all hard errors with a diagnostic message on the console.

Catastrophic errors are errors of such a severe magnitude that th program cannot continue, When a catastrophic error is detected, th program attempts to display an error message on the console terminal Following that, the processor goes into a infinite loop at IPL 31 (ther is no halt state for MicroVAX). An example of a catastrophic error i when the console program is unable to locate any working memory.

It is possible to bypass all diagnostic tasks. This may done b enabling halts and by manually halting the processor following power-u or reset. The diagnostics will be halted and the processor will ente console I/O mode. This option allows a field service engineer to bypas a failing test and enter console I/O mode where the console commands ca be used to further diagnose the problem.

As part of each diagnostic subtest, a test code is displayed on th console and on the LEDs, making it possible to monitor the progress o the diagnostics. The two display mechanisms use unrelated logic providing a high probability that at least one will be operative. If hard error is detected by a test, a diagnostic message is displayed o the console. If a catastrophic error occurs, it may not be possible t display a diagnostic message on the console but the most recent tes code will be left in the LEDs.

10 MAINTAINABILITY

The console program is located in two socketed ROM's on the KA63 processor. If necessary, they can be replaced. This is the only for of maintenance possible.

The console program displays the version number of the console progra ROM when the power up messages are displayed. This can be used t easily determine which version of the console program is present.

Upon completion of development, the source copy of the console progra will be preserved for ECO purposes by the VMS group.

11 MAINTENANCE

The console program is located in two socketed ROM's on the KA63 processor. If necessary, they can be replaced. This is the only for of maintenance possible.

12 COMPATIBILITY

12.1 Product Compatibility

This program is a new product developed for the first time for th KA630. Even though other MicroVAX based processors might be able t utilize parts of this product, no effort is planned to facilitate suc sharing beyond that which arises naturally from sound modular design.

12.2 Standards Conformance

Although not required, the command language accepted by the console i intended to be as compatible as possible with that defined in DE Standard 32. However, the following significant console features ar defined in the standard but omitted by the KA630 console program:

- MICROSTEP command - not supported by the MicroVAX chip.
- LOAD command - no console storage device is supported.
- SET command - no set options are defined.
- NEXT command - not supported by the MicroVAX chip.
- @ command - no console storage device is supported.

The console program recognizes terminal device attributes based o device attribute responses registered in DEC Standard 138.

Escape sequence parsing (used to identify terminals and to cause al escape sequences to act as line terminators) is based on the algorithm defined in the "Video System Standards Reference Manual".

12.3 Internationalization

All console program message texts are either multilingual or languag independent. The message texts displayed on power-up (shown in figure 9 and 10) are multilingual. All other texts are language independent they incorporate short language insensitive abreviations rather tha readable sentences. Depending on the user preference, the message text are output in either English, French, German, Italian, Portuguese Spanish, Dutch, Danish, Finnish, Norwegian or Swedish.

Numeric status displays on the processor LEDs facilitate the diagnosi of failing processors in a language independent manner.

The operation of the console is independent of the user's line voltag or frequency.

The console supports the Digital Multinational Character Set (MCS
This support extends to displaying foreign language messages with MC
accepting and echoing MCS characters and accepting a device attribute
report (the console queries the terminal to determine if it is a CRT
not) using the C1 control characters of MCS. However, all consol
commands must be entered using the ANSI subset.

If the terminal does not support MCS, the console uses English messa
texts.

The console program uses four characters that are national replacemen
characters, the caret (^), the backslash (\) and the right and lef
square brackets ([,]). The caret is used by the console to denot
control characters. The backslash is used to delimit text deletion
when editing console input. The square brackets are used to denot
directory specifications when the user directs the bootstrap to solici
a secondary bootstrap file name. No provision is made for terminal
which replace any of these characters.

13 EVOLVABILITY

Because the console program is in ROM and cannot be easily changed
there is little likelihood of significant evolution over its lifetime
An exception is support for additional bootstrap devices.

Support for new types of disk and tape units is provided not by th
console program itself but by the console's support for MSCP and TMSC
disks and tapes. Any new disk or tape supported by the MSCP and TMSC
controllers will be supported by the console program.

Another possible need is support for a new Ethernet controller. Any ne
controller compatible with the old will be supported. Any new an
incompatible controller could not be supported except by an ECO of th
console ROM.

The console terminal recognition code relies on compliance with DE
video terminal architecture standards to insure proper recognition o
future terminals.

14 COSTS

NOT APPLICABLE

15 TIMELINESS

This product must be available for the first release of the KA63
processor which is gated by availability of this product.

In order to insure that this goal is met, every effort will be made
minimize the interdependence of the KA630 development project a
others. For this reason, the KA630 development group will maint
complete responsibility for the entry/dispatch code and the diagnost
code. These sections are very processor implementation specific and
done outside the group, schedule risk would be raised.

The command language interpreter is less implementation specific b
because it will be needed early in the development phase of the KA63
it will be done within the development group as well.

The restart code will be done by the KA630 development group as well b
the bootstrap code will be written by the VMS development group. Th
the VMS supplied code will be available on schedule is insured by t
fact that the code is essentially the same as that in SEAHORSE, whi
will be completed previously.

16 CONSTRAINTS AND TRADEOFFS

The following list reflects the priority ordering of the goals for th
product (1 is highest):

1. Time to market - must be available for the KA630 first releas

2. Reliability - cost of field repairs is high, therefore,
errors should ship.

3. Bootstraps supported - must support RQDX1 disks and DEQ
(DECNET).

4. Diagnostic coverage - within available ROM space, diagnost
coverage should be maximized.

5. Memory consumption - ROM is available in 16, 32 and 64
amounts. The console program should attempt to meet its goa
within 32 Kb.

6. Support of QxSS based bitmapped terminals.

7. Console I/O language complexity - reduction of comma
verbosity and editing capabilities will be considered to mee
space and schedule goals.

If necessary, to meet the highest priority goals lower priority goa
will be compromised.

APPENDIX A

INTERPRETATION OF KA630 LED AND CONSOLE DISPLAY CODES

There are four red LEDs on the KA630 (echoed off the processor board a well). These four LEDs, interpreted as a four bit hexadecimal digit have the meanings described in table A-1. All LED codes are displaye during power-up. A problem is indicated only if the LED code i continuously displayed.

Each LED is illuminated if its control bit in the boot and diagnosti register (BDR) is one and is dark if its control bit is zero.

It is possible for privileged code to output to the LEDs when th processor is in program mode. Because this may cause confusion in th interpretation of the LEDs, this use of the LEDs in program mode i discouraged.

When executing the power-up sequence, the codes '9' through '0' are als output to the console terminal. See figures 9 and 10.

Table A-1: KA630 LED Interpretation

Code	Activity	Exit Criteria
F	Electrical power-up.	MicroVAX starts execution from console program ROM.
E	Wait for PWR-OK.	BDR<15> set.
D	Perform ROM checksum and TOY RAM tests. (1)	Test success.
C	Initialize console program memory.	Console memory and bitmap initialized, registers saved.
B	Run IPCR tests. (1)	Test success.
A	Test for and check out QxSS video console display if present. (1)	QxSS operational or not present.

Table A-1 (cont.)

Code	Activity	Exit Criteria
9	Perform console port tests and terminal identification. (1)	Console terminal type determined.
8	Query console language (1), then enter console command mode.	Exits automatically on power-up, otherwise exits on console CONTINUE, START, BOOT or TEST command.
7	Run memory pattern tests. (2)	At least 64 kb of contiguous good memory found.
6	Run memory address tests. (1)	All address tests passed.
5	Run I/O map tests. (1)	Test success.
4	Run CPU tests. (1)	Test success.
3	Run Interrupt tests. (1)	Test success.
2	Search for bootstrap device. (3)	Valid bootstrap device located.
1	Load bootstrap. (3)	Bootstrap successfully loaded.
0	Program mode.	Not applicable.

(1) Performed only on power-up entry into console program.

(2) Performed on power-up entry and on operator requested bootstrap.

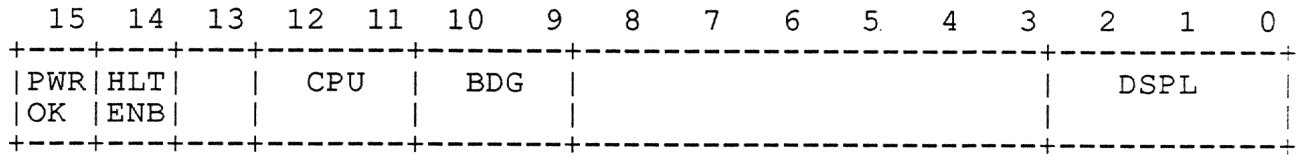
(3) Performed only during bootstrap.

APPENDIX B

SUMMARY OF I/O REGISTERS USED BY CONSOLE PROGRAM

B.1 BOOT AND DIAGNOSTIC CONFIGURATION REGISTER (BDR)

The BDR register is used to read the values of the KA630 configurati switches and to output to the processor LED display.



Boot and Diagnostic Register

Hex addr: 20080000 (hex)

BDR<15> PWR OK Power OK - set when power is stable.

BDR<14> HLT ENB Halt enable - A read only bit that enable recognition of the BHALT(L) bus signal as a hal condition and enables recognition of a BRE condition from the console terminal as a hal condition. If not set, neither condition wil result in a processor halt. This bit is set b a switch external to the KA630.

If CPMBX<1:0> is zero, this bit also control the actions of the console program in respons to any halt condition. If set the consol program will enter console I/O mode followin any halt. If not set the console program wil attempt to restart the processor and if tha fails, it will attempt to reboot the processor If that fails as well, the console program wil then enter console I/O mode. If CPMBX<1:0> i non-zero, the console program uses that field t determine its actions on processor halt.

SUMMARY OF I/O REGISTERS USED BY CONSOLE PROGRAM
BOOT AND DIAGNOSTIC CONFIGURATION REGISTER (BDR)

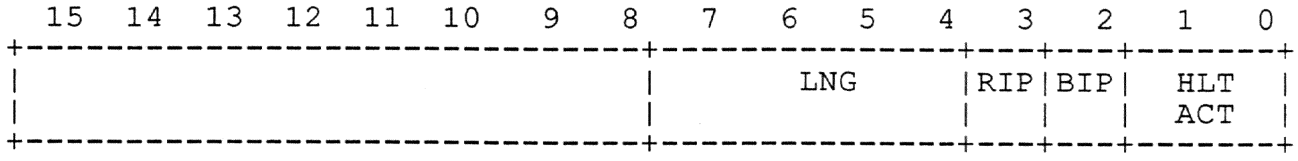
BDR<12:11>	CPU	CPU designation 0 Arbiter CPU 1 Auxiliary CPU number 1 2 Auxiliary CPU number 2 3 Auxiliary CPU number 3
BDR<10:9>	MODE	Power-up mode control bits. These two bits a combined with the HLT ENB to generate a thr bit power-up mode field. The interpretation this three bit field is given in table 1.
BDR<3:0>	DSPL	LED display bits - these write-only bits a used to load the LED status display. A b value of one causes the associated LED to illuminated, a bit value of zero causes the L to be dark.

B.2 TIME OF YEAR (TOY) CLOCK REGISTERS

The TOY registers contain the time of year clock registers as well as bytes of battery backed-up RAM. The console program references th clock registers only to check for battery backup failure. See th KA630-A Processor Specification for information on the TOY clo registers. The console program reserves all of the RAM register There are a total of 64 eight bit registers, word aligned. They may b accessed either as bytes or as words. When accessed as words, bit <15:8> are ignored on writes and return zeros on reads. The bas address of the TOY registers is 200B8000 (hex).

B.2.1 Console Program Mailbox (CPMBX)

The console program and the operating system communicate with each othe via this register.



Console Program Mailbox Register (CPMBX)

TOY register 14, Hex addr: 200B801C

CPMBX<7:4>	LNG	Console Message Text Language. This fiel

controls the output of message texts to t
console terminal. If the field is set to zer
the console program will prompt the user to s
this field on power up.

0 Prompt for language.

1	German	6	Danish	11	Portuguese
2	English	7	Dutch		
3	Spanish	8	Finnish		
4	French	9	Norwegian		
5	Italian	10	Swedish		

CPMBX<3> RIP — If set, a restart attempt is in progress. Thi
flag must be cleared by the operating system i
the restart succeeds.

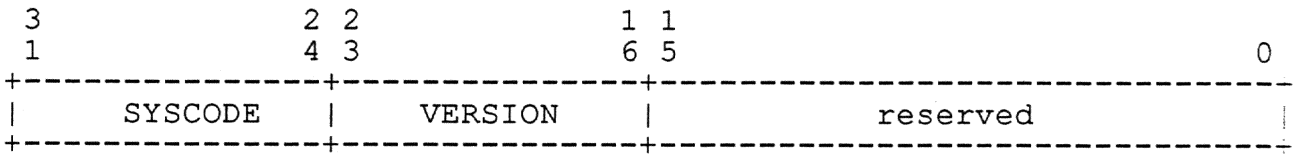
CPMBX<2> BIP — If set, a bootstrap attempt is in progress
This flag must be cleared by the operatin
system if the bootstrap succeeds.

CPMBX<1:0> HLT ACT — Processor halt action - this field is used t
control the automatic restart/bootstra
procedure. This mailbox allows operating syste
software to override the BDR HLT ENB field
Both bits are cleared on power up and when th
console program exits.

0 Use HLT ENB (BDR<14>) to determin
action.
1 Restart, if that fails, halt.
2 Reboot, if that fails, halt.
3 Halt.

B.3 SYSTEM IDENTIFICATION EXTENSION REGISTER (SIDEX)

Any MicroVAX based system must implement a longword at physical locatio
20040004 (hex) to distinguish it from other MicroVAX based system. Th
MicroVAX implements the system identification IPR but this register onl
identifies the processor implementation, not the system itself. Thi
longword exists in the physical address range of the KA630-A consol
program ROM.



System Identification Extension (SIDEX)

SIDEX<31:24> SYSCODE System code. This field is 1 for the KA630-A.

SIDEX<23:16> VERSION Version number of console program ROM.

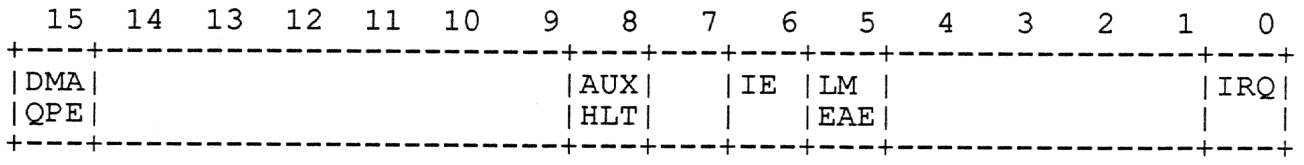
SIDEX<15:0> Reserved.

B.4 CONSOLE TERMINAL REGISTERS

The console program accesses the console terminal through four interna processor registers, the Console Receive Control and Status Registe (RXCS), the Console Receive Data Buffer Register (RXDB), the Consol Transmit Control and Status Register (TXCS), and the Console Transmi Data Buffer Register (TXDB). See the KA630-A Processor Specificatic for their definition.

B.5 INTERPROCESSOR COMMUNICATION REGISTER (IPCR)

The IPCR register is used to support communication between arbiter an auxiliary processors. It controls the visibility of auxiliary memory o the Q-22 bus, it allows an auxiliary processor to be halted by anothe processor, and it provides for an interprocessor interrupt capability The location of the IPCR in the Q-22 bus address range depends on th CPU identification code of the processor.



Interprocessor Communication Register

Hex address:	20001F40	Arbiter
	20001F42	Auxiliary #1
	20001F44	Auxiliary #2
	20001F46	Auxiliary #3

ICR<15> DMA QPE — DMA Q-22 address space parity error. This rea only bit is set whenever a parity error i detected when an external CPU or devic references KA630 local memory. (Not used by th console program).

ICR<8> AUX HLT — Auxiliary halt. A read/write bit which when se on an auxiliary processor, it causes th processor to halt. This bit has no effect i the auxiliary is in console I/O mode and i ignored all together by arbiter processors This bit is always cleared upon entry int console I/O mode. It is also used by auxiliar processors as part of its bootstrap process (se section 3.5.1.7).

ICR<6> IE — Interrupt enable. If set, an interrupt will b generated if IPCR<0> is set. To the loca processor this is a read/write bit. To externa processors this is a read only bit.

ICR<5> LM EAE — Local memory external access enable. When thi bit is set, local memory can be accessed via th Q-22 bus map. To the local processor this is read/write bit. To external processors this i a read only bit.

ICR<0> IRQ — Interrupt request. When set, a processo interrupt is requested. This bit is enabled b ICR<6>. If ICR<6> is not set, writes to thi bit are ignored and the bit is not set.

APPENDIX C

QXSS AND APT

When a KA630 system powers up for the first time it determines wh console device is present. If the console device is a QxSS vid display, it uses it as the console. However, during manufacturin systems are tested by APT which relies on the console program using t console port rather than the QxSS.

To continue to support APT testing, the KA630 console program allows AP to force the console program to revert to using the console port inste of the QxSS. To do this, APT must wait until the system has complet its power-up. APT then asserts break and the system halts. When t console program halts, it checks for the existance of a break conditi on the console port. If break is being asserted, the console progr resets itself to assume that the console port is connected to a hardco terminal and the console language is English. APT can now use t system normally.

Once this is done, there is no way short of powering the system off a turning it back on again to cause the console program to resume use o the QxSS as the console display device.

APPENDIX D

OUTSTANDING ISSUES

1. Section 1, page 1 - QVSS support needs to be further define within this specification. The intent to support is define but the exact definition of that support is TBS. This i important as QVSS is considerably more complicated to suppor than a console terminal.

2. Section 3.5.1.1, page 16 - More details on MAYA and QI descriptions are needed, planned controller names, uni designations, etc.

3. Section 4, page 38 - What should be in the publication section?

4. Section 8, page 40 - Need performance indications fro diagnostics.

5. Section 10, page 41 - Need agreement from VMS that they wil take the console program sources as part of VMS source kit.

[End of file]