EXTERNAL SPECIFICATIONS Q64

1	ER-0382-14	LUM 1-18-83	1-13-83	1-13-83	—
2	PPR-0382-20	S.L. 1/19/83	3/83	3-10-83	3/11/83
A	PR-0382-27	WP 4/25/83	—	—	—

O.B.

MAY 27 1983

"PRODUCTION RELEASE"

PROPRIETARY
NOTICE TO PERSONS RECEIVING THIS DRAWING: MDS—QANTEL, INC. CLAIMS PROPRIETARY RIGHTS TO ALL INFORMATION CONTAINED HEREIN. NEITHER THIS DRAWING, NOR ANY REPRODUCTIONS THEREOF, MAY BE USED OR DISCLOSED TO ANY THIRD PARTIES WITHOUT THE EXPRESS WRITTEN PERMISSION OF MDS-QANTEL, INC.

MATERIAL:

FINISH:

NEXT ASSEMBLY

APPROVALS		DATE
DRFTS	DLB	1/12/82
CHK		4-28-83
PROJ ENGR		4-28-83
MFG ENGR		5-11-83
PROJ MGR		5-16-83

DO NOT SCALE DRAWING

MDS QANTEL BUSINESS COMPUTERS
a Mohawk Data Sciences Company

External Specification, Q64

SIZE	DWG NO.	REV.
A	A51072-001	A

SCALE

SHEET 1 OF 9

FORM 404 04/82

1.0 INTRODUCTION

The Q64 a Central Processing Unit for standard Qantel Systems. It is a bit-slice processor based on the AMD 2901B chip. It contains ROM microcode to interpret the standard Qantel macro instruction language. It is designed to be at least 4 times faster than the Q29B.

It can interface any of the I/O controllers supported on the Q29B or Q30. Main memory capacity is 4M or 16M bytes, depending on the chip used. A new programmer's test panel will be available for debugging macro programs on the Q64.

1.1 RELATED DOCUMENTS

A52066-001	Design Specification, CPU Q64
A52067-001	Design Specification, MEM Q64A
A52069-001	Microword Specification & Assembly Language, Q64
A51073-001	Firmware Specification, Q64
A51074-001	Firmware Development System, Q64
L51067-001	Microcode Listing, Q64

2.0 MACHINE LANGUAGE COMPATIBILITY

1. All machine instructions available on the Q29B and Q30 processors are implemented on the Q64. All instructions, except where noted, operate exactly as on the Q30.
2. The Machine Identification value is $000005.
3. The Read Fast and Write Fast I/O instructions can operate at speeds up to 800 nanoseconds per byte.
4. Six new "immediate" instructions are implemented.
5. Indirect addressing is allowed to chain up to 15 levels. After 15, it is assumed to be excessive and an error condition is generated.
6. The 16 base registers are each 24 bits to address up to 16M bytes.

3.0 POWER-ON / IPL

At power-on or when the IPL button is pressed, the CPU performs the following operations:

1. CPU diagnostic tests;
2. Sets base registers to banks A/C;
3. Resets all I/O controllers (0-F);
4. Reads from device 0;
5. Branches to fetch the first Qantel instruction, usually from location 0 of main memory.

The data entered on device 0 may have several forms. If the data is all spaces, then the CPU will bootstrap to disk 0D. If the data consists of two hexadecimal characters, the CPU bootstraps to the disk with this device number (eg, 1D or 0C).

Otherwise, the data should be hexadecimal Qantel machine language instructions. The data is packed, two hex digits per byte, and placed starting at location 0 in memory. Non-hex characters (except Q and ') are ignored. Execution of Qantel code begins at location 0 (usually).

If a 'Q' is encountered, then the next 4 characters form a hexadecimal address. This address is used as the location to place the subsequent data. As a special case, if Q is the first non-blank character entered, then the hex address following the Q is used as the execution start address for the Qantel machine code.

If a quote (') is encountered, then the following data characters are interpreted as ASCII and placed directly into memory (with no packing). No characters are ignored. ASCII mode is terminated by another quote.

3.1 IPL

When the IPL Button is pressed, the current Qantel Program Counter is placed into locations 30 and 31 of memory. This is for diagnostic purposes to find where the program was executing when IPL'ed. The value may be several bytes into the current instruction.

4.0 BASE REGISTERS AND MEMORY ADDRESSING

The Q64 contains 256 24-bit base registers. They are implemented as 16 sets, each containing 16 registers. These sets are designed to be used for faster context switching. (Only set 0 is implemented at this time.) 24 bits allow addressing up to 16M bytes of main memory.

The Q64 microcode reserves the first 4K bytes of memory for its priviledged use (diagnostics, test panel interface, etc.). This is transparent to the macro code programmer. All macro instructions which load base registers automatically add $1000 to the value loaded. $1000 is subtracted when the value is read back.

As in the Q29B and Q30, the logical 15-bit address presented by the macro code consists of:

0bbb bddd dddd dddd

Upon each memory access, the 11-bit displacement is added to the contents of the 24-bit base register specified to form the 24-bit effective memory address.

5.0 ERROR CONDITIONS

If the Q64 detects an error condition during execution, it will issue a Reset I/O to all devices and write a brief error message to device 0. The messages available are:

ILLEGAL	Illegal macro instruction
INDIRECT	Excessive indirect addressing chain
MEM PAR ERR	Memory parity error

6.0 MACHINE LANGUAGE INSTRUCTION SET

The macro instruction set is downward compatible with the Q30. The instruction set is listed below with new instructions flagged by an N. The "Defined" column indicates where the instruction is defined: (R) Real Manual, (Q29) Q29B Design Spec, (Q30) Q30 Design Spec, (Q64) this document, or (Q7.5) Q7.5 Functional Description.

	Object Code		Instruction	Mnemonic	Def
		aaaa0d	Read	RD	R
	xxxxF1	aaaa0d	Read Hex	RHX	R
	ccccF2	aaaa0d	Read with Count	RDC	R
	ccccF3	aaaa0d	Read Hex with Count	RHC	R
	ccccF4	bbbb0x aaaa	CRC Calculation	CRC	Q29
	ccccF6	aaaa0d	Read Fast	RDF	Q30
	[bbbbFL]	aaaa1L	Add Decimal	ADD	R
	[bbbbFL]	aaaa2L	Subtract Decimal	SBD	R
	[bbbbFL]	aaaa3L	Multiply Decimal	MPY	R
	[bbbbFL]	aaaa4L	Divide Decimal	DIV	R
	[bbbbFL]	aaaa5L	Load Decimal	LD	R
		aaaa6L	Store Accumulator	STA	R
	bbbbFL	aaaa6L	Move	MOV	R
	[bbbbFL]	aaaa7L	Compare	CMP	R
		aaaa80	Disk Seek	SEK	R
		aaaa81 dd	Disk Bootstrap	DBS	R
		aaaa82	Load Base Register	LBR	R
	0cccFx	aaaa82 tt	Translate	TRN	R
	bbbbFL	aaaa83 LL	Search Equal	SEQ	R
	[bbbbFL]	aaaa84	Compare Numeric	CN	R
	[bbbbFL]	aaaa85	Compare Zone	CZ	R
		0dss86	Read Status 2	RS2	R
		1dss86	Read Identification	RID	Q29
	bbbbF0	aaaa86	Load Immediate	LDI	R
	bbbbF1	aaaa86	Add Immediate	ADI	R
	bbbbF2	aaaa86	Subtract Immediate	SBI	R
	bbbbF3	aaaa86	Move Immediate	MVI	Q29
N	bbbbF4	aaaa86	And Immediate	ANI	Q64
N	bbbbF5	aaaa86	Or Immediate	ORI	Q64
N	bbbbF6	aaaa86	Xor Immediate	XRI	Q64
N	bbbbF7	aaaa86	Test Bit Immediate	TBI	Q64
N	bbbbF8	aaaa86	Compare Immediate, 1 byte	CP1	Q64
N	bbbbF9	aaaa86	Compare Immediate, 2 bytes	CP2	Q64
		aaaa87	Read Base Register	RBR	R
	ccccF0	bbbb87 aaaa	Translate Right	TR	R
	ccccF2	bbbb87 aaaa	Scan Unequal Right	SNR	R
	ccccF3	bbbb87 aaaa	Scan Equal Right	SER	R
	ccccF4	bbbb87 aaaa	Fill Right	FR	R
	ccccF5	bbbb87 aaaa	Move Right	MR	R

Object Code			Instruction	Mnemonic	Def
ccccF6	bbbb87	aaaa	Exchange Right	XR	R
ccccF7	bbbb87	aaaa	Compare Right	CR	R
ccccF8	bbbb87	aaaa	Translate Left	TL	R
ccccF9	bbbb87	aaaa	Hex to Ascii Conversion	HAC	Q29
ccccFA	bbbb87	aaaa	Scan Unequal Left	SNL	R
ccccFB	bbbb87	aaaa	Scan Equal Left	SEL	R
ccccFC	bbbb87	aaaa	Fill Left	FL	R
ccccFD	bbbb87	aaaa	Move Left	ML	R
ccccFE	bbbb87	aaaa	Exchange Left	XL	R
ccccFF	bbbb87	aaaa	Compare Left	CL	R
	aaaa88		Scan Interrupt	SCI	R
bbbbF0	aaaa88		Load Base Register & Branch	LBB	R
	ssss89		Return from Subroutine	RET	R
ssssF0	aaaa89		Subroutine Call	SUB	R
ssssF1	aaaa89		Pull Data	PUL	R
ssssF2	aaaa89		Push Data	PSH	R
ssssF3	aaaa89		Pop Data	POP	R
	aaaa8A		Increment by 1	IN1	R
ttttF0	aaaa8A		Branch on Table	BT	Q29
bbbbF1	aaaa8A		Loader Relocation	LDR	Q29
	aaaa8B		Decrement by 1	DC1	R
ccccF0	ssss8B	1111	Load Base List	LBL	Q29
ccccF1	ssss8B	1111	Read Base List	RBL	Q29
	aaaa8C		Increment by 2	IN2	R
	aaaa8D		Decrement by 2	DC2	R
	aaaa8E		Increment by 3	IN3	R
bbbbFL	aaaa8E		Get Parameters	GPT	Q29
	aaaa8F		Decrement by 3	DC3	R
bbbbFL	aaaa8F		Base Register Convert	BRC	Q29
[bbbbFx]	aaaa90		And	AND	R
[bbbbFx]	aaaa91		Or	OR	R
[bbbbFx]	aaaa92		Exclusive Or	XOR	R
[bbbbFx]	aaaa93		Test Bit	TBT	R
[bbbbFL]	aaaa94		Move Numeric	MN	R
[bbbbFL]	aaaa95		Move Zone	MZ	R
[bbbbFx]	aaaa96		Compare Decimal	CD	R
	aaaa97		Shift Bit Left	SBL	R
ccccFL	aaaa97		Shift Field Left	SHL	R
	aaaa98		Shift Bit Right	SBR	R
ccccFL	aaaa98		Shift Field Right	SHR	R
[bbbbFL]	aaaa99		Edit	EDT	R
bbbbFL	aaaa9A		Unedit	UED	R
[bbbbFL]	aaaa9B		Pack	PAK	R
[bbbbFL]	aaaa9C		Unpack	UPK	R
	0dss9D		I/O Execute	IOX	Q7.5
	1dss9D		Device Control	CTL	R
	2dss9D		Set Read	SRD	R
	4dvv9D		Status In	SIN	R
	xdxx9E		Reset I/O	RIO	R
	xxxx9F		Return from Interrupt	RTI	R
xxxxF0	xxxx9F		Disable Interrupts	DI	Q29

Object Code		Instruction	Mnemonic	Def
xxxxF1	xxxx9F	Enable Interrupts 1	EI	Q29
xxxxF2	xxxx9F	Enable Interrupts 2	EI2	Q29
	aaaaA0	No Operation	NOP	R
	aaaaA1	Branch on Overflow	BOV	R
	aaaaA2	Branch on Minus	BMI	R
	aaaaA3	Branch on Nonzero	BNZ	R
	aaaaA4	Branch on Zero	BZ	R
	aaaaA5	Branch on Not Minus	BNM	R
	aaaaA6	Branch on No Overflow	BNO	R
	aaaaA7	Branch Unconditional	BRU	R
	aaaaA8	Halt & Branch	HLT	R
	aaaaA9	Branch and Link	BLI	R
	aaaaAA	Branch on Plus	BP	R
	xxxxAC	Set Bank C	BKC	R
	xxxxAD	Set Bank D	BKD	R
	xxxxAE	Set Bank E	BKE	R
	aaaaAF	Branch on Not Plus	BNP	R
	aaaaBd	Write	WR	R
xxxxF1	aaaaBd	Write Hex	WHX	R
ccccF2	aaaaBd	Write with Count	WRC	R
ccccF3	aaaaBd	Write Hex with Count	WHC	R
ccccF4	aaaaBd	Read Check	RCH	Q29
ccccF6	aaaaBd	Write Fast	WRF	Q30
[bbbbFL]	aaaaCL 02	Load Binary	LDB	Q29
[bbbbFL]	aaaaCL 03	Multiply Binary	MPB	Q29
[bbbbFL]	aaaaCL 04	Divide Binary	DVB	Q29
[bbbbFL]	aaaaCL 05	Binary to Decimal Conversion	BDV	Q29
[bbbbFL]	aaaaCL 06	Decimal to Binary Conversion	DBV	Q29
	aaaaCx 07	Machine Identification	MID	Q29
bbbbFm	aaaaCL 07	Key Search	KSR	Q29
	cccc ddee			
[bbbbFL]	aaaaDL	Add Binary	ADB	R
[bbbbFL]	aaaaEL	Subtract Binary	SBB	R

6.1 NEW INSTRUCTION SUMMARY

ANI And Immediate
ORI Or Immediate
XRI Xor Immediate
TBI Test Bit Immediate
CP1 Compare Immediate, 1 byte
CP2 Compare Immediate, 2 bytes

The new instructions are described on the following pages.

6.2 NEW INSTRUCTION DESCRIPTIONS

6.2.1 AND IMMEDIATE

ANI value;address bbbbF4 aaaa86

This instruction performs a logical 1-byte AND between the value in the A-operand with the contents of the B-operand address. The resulting byte is placed in the B field. The Nonzero flag is set according to the result of the operation.

6.2.2 OR IMMEDIATE

ORI value;address bbbbF5 aaaa86

This instruction performs a logical 1-byte OR between the value in the A-operand with the contents of the B-operand address. The resulting byte is placed in the B field. The Nonzero flag is set according to the result of the operation.

6.2.3 XOR IMMEDIATE

XRI value;address bbbbF6 aaaa86

This instruction performs a logical 1-byte exclusive OR between the value in the A-operand with the contents of the B-operand address. The resulting byte is placed in the B field. The Nonzero flag is set according to the result of the operation.

6.2.4 TEST BIT IMMEDIATE

TBI value;address bbbbF7 aaaa86

This instruction performs a logical 1-byte Test Bit (AND) between the value in the A-operand with the contents of the B-operand address. The Nonzero flag is set according to the result of the operation. The B field is not modified.

6.2.5 COMPARE IMMEDIATE 1 BYTE

CP1 value;address bbbbF8 aaaa86

The 1-byte value specified as the A-operand is compared with the contents of the B-operand field. The comparison is for exactly one byte. The Nonzero flag is set if they are not equal. The Minus flag is set if the A-operand value is greater than B's contents.

6.2.6 COMPARE IMMEDIATE 2 BYTES

CP2 value;address bbbbF9 aaaa86

The 2-byte value specified as the A-operand is compared with the contents of the B-operand field. The comparison is for two bytes. The Nonzero flag is set if they are not equal. The Minus flag is set if the A-operand value is greater than B's contents.