CONTENTS

USER GUIDE

Disclaimer

Every effort has been made to test this product and its operation.

DTM reserves the right to make changes in specifications at any time and without notice. The information in this document is believed to be accurate, but no responsibility is assumed for its use, nor for any infringements of patents or other rights of third parties resulting from its use.

INSTALLING THE M200

The M200 should first be checked for any signs of visible damage which may have resulted from transportation.

It is advisable to read the user guide fully before applying power to the M200.

The M200 has to be fitted to the connector slots inside the PC machine. This requires the removal of the outer cover, normally achieved by removing some screws from the rear and sliding the outer cover forward exposing the main circuit board of the PC.

Installation requires removing a blanking plate from the rear face of the machine, this provides a free slot for the M200 to be installed in. It is important to note that the M200 takes up 2 slots of space inside the machine and that the blanking plate should be removed from a slot which has a free slot to its right as viewed from the front of the machine.

The M200 simply plugs into the free slot by using minimal pressure from the top, be carefull not to use excessive force to push the board home as this could jar the hard disk or cause other forms of damage. Ensure that the trailing edge of the M200 is securely fitted into the card guide for it.

The M200 is secured using the screw which was holding the blanking plate in place. As the M200 is a fairly heavy item it is recommended that the retaining screw be used at all times and not forgotten about.

The main circuit board derives power directly from the edge connector.

***** The hard disk derives its power separately from one of the 4 drive power connectors fitted to the machine, this typically is a white plastic 4 way connector which fits into the similar mating half which is located on the M200 hard disk unit. This connector is polarised and can only be fitted one way round, do not use excessive force to fit this connector as this may indicate that the connector is the wrong way round. Should your machine not have a spare drive power connector then contact your supplier.**

The unit can now be tested as described in the user guide. It is recommended that the machine that it is fitted to is not run with the cover off as this allows foreign matter entry to the machine which can cause damage.

USER GUIDE

1 Introduction

The QMS M200 is a four transputer module motherboard for IBM XT and AT type machines. It has a disk interface circuit built onto the motherboard. The disk interface is controlled by the M212 transputer which has one of its two INMOS links attached to a link adaptor which in turn is present on the IBM I/O bus, the second link of the M212 is brought to the 'D' type connector together with another link adaptor link, which is also connected to the IBM bus, and ten links from the four module sites. The four module sites are connected in a default pipeline using links 2 and 1, with all the links 0 and 3 together with the pipehead and pipetail brought to the 'D' type to allow simple network configuration to be done by hardwire at the plug in to the 'D' connector.

The interface to the IBM bus is used to allow data to pass to and from both the M212 disk interface and also to the transputer modules, the interface has full DMA and Interrupt capability to allow quick and efficient data transfer.

2 Hardware Description

2.1 Disk system

The Disk interface is driven directly from the M212 transputer and is a standard ST506 interface, which will directly drive the single 3.5" winchester disk provided with the M200, but it is capable of driving up to two disks which can be remotely sited and therfore physically larger when used with a simple buffer board (QMS M201).

2.2 M212

The M212 transputer uses its two links to act as a disk controller capable of recieving commands from two sources. The hardware configuration determines these two sources: the IBM system via a link adaptor onto the IBM bus, and the transputer module system. This means that the transputer domain can access the disk system directly without having to go through the IBM disk system, this allows much faster I/O for the transputer system to disk. The kinds of improvement in performance possible are given in the following examples :

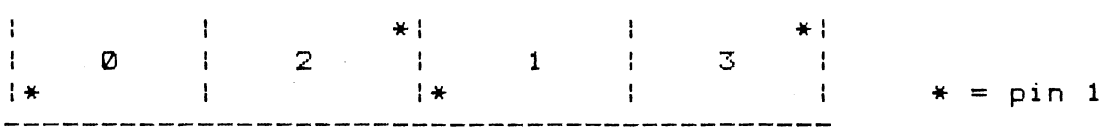
a) A standard transputer board using a software polled link adaptor achieves transfer rates of about 30Kbytes/sec, this takes over two seconds to load a 64Kbyte file. b) A transputer module connected directly to the M212 over a 20Mhz link will pass data at 800KBytes/sec, this typically exceeds the capability of the disk to supply data and consequently the time taken is now limited by the disk. If the file is contiguous on the disk then a 64K file is about 8 tracks of data which will probably also require a couple of step commands to read all 8 tracks, the total time for this to be read by the M212 with an interleave of 2 would be about 150ms and with an interleave of 1 only 75ms.

The overall improvement in performance to a system depends upon how much disk I/O is actually done. In systems which are disk intensive the benefits can become significant, without having to use software buffering techniques which can still be implimented at a future date to enhance performance. If the data on the disk is severely fragmented requiring a lot of head stepping to recover the data then the possible benefits will not be as easilly seen because the disk unit will be spending more time stepping to the data than transfering the data.

To release the full potential outlined above special software is required to run on the M212 which will interpret disk filing commands as generated by applications programs and operate on them in the same way as the IBM operating system would. This requires the special software to operate as both a disk filing system and a protocol converter, QTM is producing some simple software products for the disk filing operation which will work with some of the currently used server system protocols. It is not our intent to support every different protocol variation, users would have to program at this level themselves.

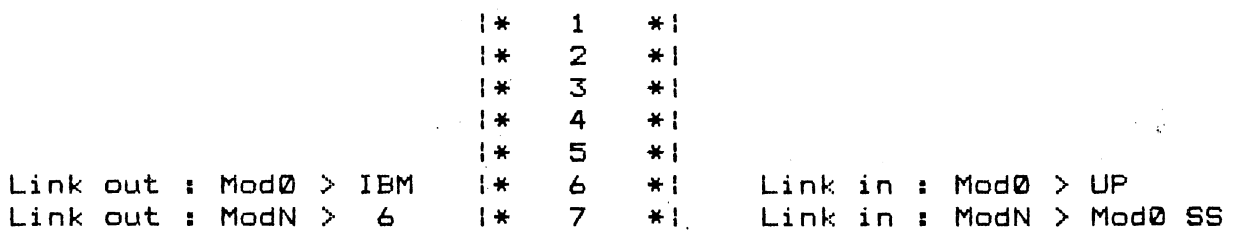
2.3 The module system

The M200 motherboard has 4 Transputer module (TRAM) sites which can accomodate a maximum of 4 TRAMs. The module sites are organised :



Module site 0 has 3 extra pins which are used by module 0 to generate reset signals to other modules, These are known as the subsystem pins. The module system has a variety of reset configurations, they are most easily defined with 2 signals : firstly module 0 can be reset from the IBM or from the external world via the UP set of pins on the 'D' type, secondly the rest of the modules can be reset from either module 0 SUBSYSTEM or from the signal chosen in the first step above.

These selections are achieved by using jumper plugs on the 2 way connector mounted adjacent to the 'D' connector.



These jumper plugs are easilly obtainable, or can be supplied with the unit.

When modules are fitted it is sometimes necessary to jumper over the default pipeline connections, eg if only modules 0 and 1 are fitted then jumpers at the sites for modules 2 and 3 are needed to bring link 2 of module 1 out to pipetail. These jumpers are simply connections between link2 and link1 in and out pins, they can easily be made, or they can be supplied with modules or motherboards.

2.4 The IBM interface

The IBM interface consists of 3 elements, the BIOS EPROM which sits at $D000X in the IBM memory map and is 32KBytes long, and 2 Link Adaptor systems.

The Link adaptor systems are for the M212 disk interface and the Module control interface. The I/O address range that these devices occupy are selectable by jumpers (see below). It is possible to select one of four different sources of interrupt event which the transputer domain can signal to the IBM domain, these are:

1) Interrupt on end of DMA transfer
2) Interrupt on transputer error
3) Interrupt on link data input ready
4) Interrupt on link data output ready

The register addressing in I/O space is :

L/A 1 at address $300 or $250	L/A 2 at address $150 or $200
Data in @ L/A 1 + $00	Data in @ L/A 2 + $00
Data out @ L/A 1 + $01	Data out @ L/A 2 + $01
I/P Stat @ L/A 1 + $02	I/P Stat @ L/A 2 + $02
O/P Stat @ L/A 1 + $03	O/P Stat @ L/A 2 + $03
Reset (W)@ L/A 1 + $10	Reset (W)@ L/A 2 + $10
Error (R)@ L/A 1 + $10	Error (R)@ L/A 2 + $10
Analyse @ L/A 1 + $11	Analyse @ L/A 2 + $11
DMA @ L/A 1 + $12	DMA @ L/A 2 + $12
IRQ @ L/A 1 + $13	IRQ @ L/A 2 + $13

The function of the unique registers is given by:

Reset register (a WRITE ONLY register)

Writing bit 0 : to '1' asserts Reset : to '0' deasserts

Error register (a READ ONLY register)

Reading bit 0 : as '1' indicates Error : as '0' no Error

Analyse register (a WRITE ONLY register)

Writing bit 0 : to '1' asserts Analyse : to '0'deasserts

DMA request register (a Read/Write register)

Writing bit 0 : as '0' initiates DMA FROM IBM TO M200
as '1' initiates DMA TO IBM FROM M200

IRQ control register (a Read/Write register)

bit 0 : '1' enables '0' disables : IRQ on DMA end
bit 1 : '1' enables '0' disables : IRQ on Error
bit 2 : '1' enables '0' disables : IRQ on Data Rx ready
bit 3 : '1' enables '0' disables : IRQ on Data Tx ready

3 Software

The M200 is provided with software written in 8086 assembly language to allow the IBM interface to the disk to operate. This software is provided in the EPROM on the main board and is also available on disk. This software is the MS-DOS device driver which allows MS-DOS to access the hard disk. The device driver provided will typically show a 20% improvement in performance over a standard hard disk in an 8086 machine, but benchmarks at about the same level as a standard AT disk drive.

The transputer software which can be run on the M212 to provide fast transputer disk access are separate products, these can be supplied in a replacement EPROM which needs to be selected for the particular protocol desired, should a range of different protocols be desired then diferent software modules can be provided on disk to allow the user to tailor transputer code to individual server variants. Please check with your supplier for further details on any of these options.

3.1 Testing the board

Included with the unit are separate routines which allow the user to test the functionality of the board, these test for the Link conections, the DMA interface and the Interrupt interface on both the M212 and the module(s) (if module(s) are fitted), the unit as supplied will be pre-formatted to MS-DOS standard and should work from power up. All the test software is self documenting and should function automatically , the user has to select which DMA and IRQ channels are in use as well as the addresses of the link adaptors if changed from the default.

4 Configuration

4.1 Jumper setting

There are 8 jumper selectors on the main board, they are located close to the M212 transputer as shown. They select as indicated :

		link out	link in
ABt	1	off	(auto-boot from winchester)
ItR	2	(Mode1 M212 ROM on)	off
BtL	3	(Boot from Mode1 ROM)	off (ie boot link)
A3	4	}	
A2	5	} see below	
A1	6	}	
10M	7	20Mhz links	10Mhz links
M2c	8	M2 control via IBM	(M2 control external)

The addressing of the link adaptors follows the following table :

	A1	A2	A3	ladp M2	ladpmod0
	0	0	0	-	-
	0	0	1	-	$200
1 = lnk out	0	1	0	$250	-
	0	1	1	$250	$200
0 = lnk in	1	0	0	$300	$200
	1	0	1	-	$150
	1	1	0	$300	-
	1	1	1	$300	$150

The selection of DMA channel is also done with jumpers :

```
1      DRQ      2  1      DAK      2

*       *       *  *       *       *  No DMA selection
*-------*       *  *-------*       *  Selects DMA 1
*       *-------*  *       *-------*  Selects DMA 2
```

DMA chan 1 is nominally free but is often used for ethernet cards, DMA chan 2 is the floppy disk DMA chan which can be disabled by writing to I/O location $3F2, allowing it to be used by the M200 for data transfer. Whenever a diskette access is required the disk bios always enables location $3F2 this means that the M200 driver must disable location $3F2 and enable its drivers at the start of a DMA transfer and then disable its drivers at the end of a transfer. Separate software routines are provided for handling the diferent DMA channels.

The selection of IRQ channel is done with a jumper selection :

```
3      IRQ      6

*       *       *  No IRQ Selected
*-------*       *  Selects IRQ 3
*       *-------*  Selects IRQ 6
```

IRQ channel 3 is nominally given over to the second RS232 port, while channel 6 is associated with the diskette driver . The software routines for DMA channel 1 support IRQ 3 and the

routines for DMA channel 2 support IRQ 6.

5 Software listings

These software listings are designed to provide a brief overview of the kind of techniques which can be used to program the M200 IBM interface.

```
CONST
                                        { M200 REGISTERS }
     boardbase      :      INTEGER = $0300 ;
     inputData      :      INTEGER =  0    ;
     outputData     :      INTEGER =  0    ;
     inputStatus    :      INTEGER =  0    ;
     outputStatus   :      INTEGER =  0    ;
     resetM2        :      INTEGER =  0    ;
     analyseM2      :      INTEGER =  0    ;
     DMArequest     :      INTEGER =  0    ;
     INTenable      :      INTEGER =  0    ;


     writeDMA   = 0  ;
     readDMA    = 1  ;
     IBMtoM200  = 8  ;
     M200toIBM  = 4  ;
     chan1      = 1  ;

     DMAchannel     :      INTEGER  = chan1 ;


     DMAstatus       = $08 ;
     DMAcommand      = $08 ;
     softDRQ         = $09 ;
     DMAsingmask     = $0A ;
     DMAmode         = $0B ;
     DMAffclear      = $0C ;
     DMAmastclr      = $0D ;
     DMAallmask      = $0F ;
     addrch1         = $02 ;
     wordch1         = $03 ;
     pagech1         = $83 ;


     PICbase        = $20 ;
     spurioustuff   = 7   ;
     DMAInt         = 1   ;

     Intchannel      :  BYTE   = 3    ;
     intmask         :  BYTE   = $F7  ;
     intvec          :  BYTE   = $28  ;
     eoi             :  BYTE   = $20  ;
     flagint         :  BOOLEAN = FALSE ;
     intcount        :  INTEGER = 0     ;
     intdata         :  BYTE    = 0     ;

VAR
    teststring  : STRING [ 200 ];
    answer      : STRING [10 ] ;
    datablock   : ARRAY [ 0..511] OF CHAR ;
```

```
    transfers    : INTEGER ;
    counter      : INTEGER ;
    blocklength  : INTEGER ;


PROCEDURE initconst ;
BEGIN
     inputData       := boardbase  ;
     outputData      := boardbase + 1  ;
     inputStatus     := boardbase + 2  ;
     outputStatus    := boardbase + 3  ;
     resetM2         := boardbase + $10;
     analyseM2       := boardbase + $11;
     DMArequest      := boardbase + $12;
     INTenable       := boardbase + $13;

     intmask         := $F7 ;
     intvec          := $28 ;
     eoi             := $20 ;

END ;

PROCEDURE IntDMA ;          { Int routine exec on DMA end Interrupt }
BEGIN
     INLINE ($50/$53/$51/$52/$56/$57/$1E/$06/$FB) ;
     flagint  := TRUE ;              { flag that DMA has ended       }
     PORT [ INTenable ] :=0   ;      { toggle Intenable to clear Int }
     PORT [ PICbase ]   :=eoi ;
     INLINE ($07/$1F/$5F/$5E/$5A/$59/$5B/$58/$CF) ;
END ;


PROCEDURE initINT ;

VAR
   IntServAddr    :  INTEGER  ;
   oldmask        :  BYTE     ;

BEGIN
     IntServAddr  := OFS (IntDMA) + spurioustuff ;

     MEM [ $0000:intvec + 0] := IntservAddr AND $FF ;
     MEM [ $0000:intvec + 1] := IntServAddr SHR 8   ;
     MEM [ $0000:intvec + 2] := CSEG AND $FF ;
     MEM [ $0000:intvec + 3] := CSEG SHR 8   ;

     oldmask  := PORT [ PICbase + 1] ;
     PORT [PICbase +1]  := oldmask AND intmask ;
     PORT [INTenable]  := 0 ;

END ;

PROCEDURE initC012 ;
BEGIN
        PORT [  inputStatus ]  := 2  ;  { Enable inputInt }
        PORT [ outputStatus ]  := 2  ;  { Enable inputInt }
END ;

FUNCTION dataPresent  : BOOLEAN ;
BEGIN
```

```
        dataPresent   := ODD ( PORT [ inputStatus ] ) ;
END;


FUNCTION outputReady   : BOOLEAN ;
BEGIN
        outputReady := ODD ( PORT [ outputStatus ]) ;
END;'


PROCEDURE outByte ( b : BYTE ) ;
BEGIN
        PORT [ outputData ] := b ;
        WHILE NOT ODD ( PORT [ outputStatus ] ) DO ;
END ;

FUNCTION inByte : INTEGER  ;
BEGIN
        WHILE NOT ODD ( PORT [inputStatus] ) DO ;
        inByte := PORT [ inputData ] ;
END;

PROCEDURE loopFor   ( i   : INTEGER ) ;
BEGIN
        WHILE  i <> 0  DO  i := i-1  ;
END ;


PROCEDURE doReset ;
BEGIN
        PORT [ analyseM2 ] := 0 ;
        loopFor ( 800) ;
        PORT [ resetM2   ] := 1 ;
        loopFor (3000) ;
        PORT [ resetM2   ] := 0 ;
        loopFor (1000) ;
END ;

PROCEDURE clearError ;
BEGIN
        doReset ;
        initC012 ;
        outByte (4) ;
        outByte ($22) ; outbyte ($F9) ; outByte ($60) ; outByte ($0E) ;
END ;


PROCEDURE SetError ;
BEGIN
        doReset ;
        initC012 ;
        outByte (6) ;
        outByte ($22) ; outByte ($F9) ; outByte ($25) ;
        outByte ($F8) ; outByte ($21) ; outByte ($F0) ;
END ;



PROCEDURE loadT2code ;
VAR
```

```
        data     : BYTE ;
        bootcode : FILE OF BYTE ;
BEGIN
        ASSIGN ( bootcode , 'seclis.b2' ) ;
        RESET  ( bootcode ) ;
        WRITELN ( 'Loading bootcode to transputer ... ') ;
        REPEAT
                READ ( bootcode , data ) ;
                outByte ( data ) ;
        UNTIL EOF ( bootcode ) = TRUE ;
        WRITELN ( ' Loaded code ' ) ;
END ;

PROCEDURE add24bit (segment,offset : INTEGER ; VAR byte0,byte1,byte2 : BYT
VAR
        result  : INTEGER ;
        temp    : INTEGER ;
        a       : BYTE ;
        b       : BYTE ;
        carry   : BYTE ;

BEGIN
        temp    := segment SHL 4 ;
        a       := temp AND $FF  ;
        b       := offset AND $FF ;
        result  := a + b ;
        byte0   := result AND $FF ;
        carry   := result SHR 8 ;

        a       := temp SHR 8 ;
        b       := offset SHR 8 ;
        result  := a + b + carry ;
        byte1   := result AND $FF ;
        carry   := result SHR 8 ;

        temp    := segment SHR 12 ;
        byte2   := temp + carry ;
END ;


PROCEDURE setDMA (mode:BYTE; length,segdata,ofsdata:INTEGER); EXTERNAL 'se

PROCEDURE setDMAup (mode:BYTE;length,segdata,ofsdata:INTEGER); EXTERNAL 's

PROCEDURE setupDMAC ( readnotwrite : BYTE ; length,segdata,ofsdata : INTEG
    VAR
        addr0 , addr1 , addr2 , direction : BYTE ;
    PROCEDURE addressandcount ( addrch, pagech, wordch : BYTE ) ;
    BEGIN
        PORT [ addrch ] := addr0 ;
        PORT [ addrch ] := addr1 ;
        PORT [ pagech ] := addr2 ;
        PORT [ wordch ] := length AND $FF ;
        PORT [ wordch ] := length SHR 8 ;
    END ;
BEGIN
        PORT [ DMAsingmask ] := 4 OR DMAchannel  ;
        PORT [ softDRQ ] := 0 OR DMAchannel ;
        PORT [ DMAffclear ] := 0 ;
        add24bit (segdata , ofsdata , addr0 , addr1 , addr2 ) ;
```

```
        addressandcount ( addrch1 , pagech1 , wordch1 ) ;
        PORT [ DMAmode ]     := ( $40 OR readnotwrite OR DMAchannel ) ;
        PORT [ DMAcommand ] := $00 ;
        PORT [ DMAsingmask ] := 0 or DMAchannel ;
END ;

PROCEDURE pollDMAC ;
VAR
        chanmask : BYTE ;
BEGIN
        chanmask := 1 SHL DMAchannel ;
        WRITELN (' Polling DMA controller ... ') ;
        WHILE (( PORT [ DMAstatus ] AND chanmask ) = 0 ) DO ;
        WRITELN (' Finished polling ' ) ;
        WRITELN ;
END ;

PROCEDURE pollIntflag ;
BEGIN
        WRITELN ('Waiting for interrupt ....') ;
        WHILE flagint = FALSE DO ;
        WRITELN (' Interrupt complete ') ;
        WRITELN ;
END ;


PROCEDURE DMAwrite ;
VAR
        wtmode  : BYTE  ;
BEGIN
        WRITELN (' Transfering data to M200 ') ;
        wtmode   := $40 OR IBMtoM200 OR DMAchannel ;
        setDMA ( wtmode , 1 ,SEG(datablock[0]),OFS(datablock[0])) ;
   {     PORT [ DMArequest ] := writeDMA ;    }
   {     pollDMAC ;                           }
END;

PROCEDURE DMAread ;
VAR
        rdmode          :  BYTE    ;
        strcount        :  INTEGER ;
        msgbackarr      :  ARRAY [0..511] OF CHAR ;
BEGIN
        WRITELN (' Reading data from M200 ') ;
        rdmode := $40 OR M200toIBM OR DMAchannel ;
        setDMA ( rdmode , 1 , SEG(msgbackarr[0]),OFS(msgbackarr[0])) ;
    {    PORT [ DMArequest ] := readDMA ; }
    {    pollDMAC ;                       }
        WRITELN (' Message received is : ' ) ;
        FOR strcount := 0 to 511  DO WRITE  ( msgbackarr[strcount]) ;
END ;

PROCEDURE switches ;
BEGIN
        boardbase := $0300 ;
        DMAchannel:= 1 ;
        intchannel:= 3 ;
END ;
```

```
BEGIN
        WRITELN ( ' Starting ... ') ;
        switches ;
        initconst ;
        initINT ;
        transfers := 0 ;
        doreset ;
        initC012 ;
        WRITELN (' o/p T2 code ? Y/N ') ;
        READLN ( answer ) ;
        IF ( answer = 'Y') OR ( answer = 'y') THEN loadT2code ;

{        FOR transfers := 1 TO 2 DO     }
        BEGIN
                WRITELN ('------------------------------') ;
                WRITELN ;
                FOR counter := 0 to 511 DO datablock [counter] := 'U' ;
              {  FOR counter := 0 to 511 DO WRITE (datablock[counter]) ; }
              {  WRITELN ('type in a teststring') ; }
              {  READLN (teststring) ;               }
                teststring := 'This is a DMA test and this is another test
                WRITELN ;
                FOR counter := 0 to (LENGTH (teststring) -1 ) DO
                    BEGIN
                         datablock[counter]:=teststring[counter+1]  ;
                    END ;
                datablock [509] := '3' ;
                datablock [510] := '2' ;
                datablock [511] := '1' ;
                FOR counter := 0 to 511 DO WRITE (datablock[counter]) ;
                WRITELN ;
           {     WHILE TRUE DO        }
           {         BEGIN            }
                       outbyte (1) ;
                       DMAwrite  ;
                       WRITELN ('dma write complete : starting read');
           {            outbyte (11) ;  }
           {         END ;              }
                outbyte (0) ;
                DMAread  ;
                WRITELN ('dma read complete') ;
            {    outbyte (0) ;             }
            {    DMAread ;                 }
            {    WRITELN ('and again') ;   }
        END ;
END.
```

6 PAL Equations

These PAL equations are intended to provide insight into the board operation and supliment the logic diagram shown below.

```
NAME            IC1v            ;
REVISION        01              ;
DATE            21/12/87        ;
DESIGNER        ARG             ;
COMPANY         QTM (Design) Ltd;
ASSEMBLY        m200            ;
DEVICE          p22v10          ;  /* D0..3 registered to provide fback */


/* CLOCKS */
        PIN 1   =  Pclk             ;       /* register clock       */

/* INPUTS */
        PIN 2   =  InIntM2LADP      ;       /* input (a)   */
        PIN 3   =  OutIntM2LADP     ;       /* output(a)   */
        PIN 4   =  InIntMOD0LADP    ;       /* input (b)   */
        PIN 5   =  OutIntMOD0LADP   ;       /* output(b)   */
        PIN 6   =  D3               ;
        PIN 7   =  D2               ;
        PIN 8   =  D1               ;
        PIN 9   =  D0               ;
        PIN 10  = !EndDMA           ;       /* ic3 p18 */
        PIN 11  = !WriteInt         ;       /* ic2 p23 */

/* OE */
        PIN 13  = !ReadInt          ;       /* ic2 p15 */

/* INPUTS/OUTPUTS */
        PIN 14  = !ErrorfromMOD0    ;   /*  I       ic6 p17
        PIN 15  =  ErrorfromM2      ;   /*  I
        PIN 16  =  IRQ              ;   /*  O
        PIN 17  =  D0_DMASel        ;   /*  I/O  D  registered for const fbac
        PIN 18  =  D1_ErrSel        ;   /*  I/O  D  registered for const fbac
        PIN 19  =  D2_OutSel        ;   /*  I/O  D  registered for const fBac
        PIN 20  =  D3_InSel         ;   /*  I/O  D  registered for const fBac
        PIN 21  =  IRQenable        ;   /*  O/I needed for pterms
        PIN 22  =             nc2   ;   /*
        PIN 23  =             nc1   ;   /*  I

FIELD DataOut. = [D3_InSel,D2_OutSel,D1_ErrSel,D0_DMASel]   ;

D0_DMASel.d =   D0 & WriteInt
            £ D0_DMASel & !WriteInt                                     ;
D0_DMASel.ar='b'0 ; D0_DMASel.sp='b'0 ; D0_DMASel.oe=ReadInt            ;
D1_ErrSel.d =   D1 & WriteInt
            £ D1_ErrSel & !WriteInt                                     ;
D1_ErrSel.ar='b'0 ; D1_ErrSel.sp='b'0 ; D1_ErrSel.oe=ReadInt            ;
D2_OutSel.d =   D2 & WriteInt
            £ D2_OutSel & !WriteInt                                     ;
D2_OutSel.ar='b'0 ; D2_OutSel.sp='b'0 ; D2_OutSel.oe=ReadInt            ;
D3_InSel.d  =   D3 & WriteInt
            £ D3_InSel & !WriteInt                                      ;
D3_InSel.ar ='b'0 ; D3_InSel.sp ='b'0 ; D3_InSel.oe =ReadInt            ;
                                        /* DataOut.oe  = !ReadInt
```

```
                                                /*    EndDMA     =   TC & DACK

IRQ         =    (InIntM2LADP £ InIntMODØLADP) & D3_InSel
               £ (OutIntM2LADP £ OutIntMODØLADP) & D2_OutSel
               £ (ErrorfromM2 £ ErrorfromMODØ) & D1_ErrSel
               £ (EndDMA £ IRQ) & DØ_DMASel                              ;

IRQenable   = D3_InSel £ D2_OutSel £ D1_ErrSel £ DØ_DMASel ;
IRQ.oe      = IRQenable                                                  ;
```

```
NAME            IC2v            ;
REVISION        01              ;
DATE            21/12/87        ;
DESIGNER        ARG             ;
COMPANY         QTM (Design) Ltd;
ASSEMBLY        m200            ;
DEVICE          p22v10          ;   /* 22v10 for external M2 res/ana/err *

/* CLOCKS */
        PIN 1   =  Pclk                         ;

/* INPUTS */
        PIN 2   = !notIOW                       ;
        PIN 3   = !notIOR                       ;
        PIN 4   = !notSYS                       ;   /* ic4 p17 */
        PIN 5   =  A1                           ;
        PIN 6   = !notSYSMOD0                   ;   /* ic4 p16 */
        PIN 7   =  A0                           ;
        PIN 8   = !notSYSM2                     ;   /* ic4 p15 */
        PIN 9   =                   nc1         ;
        PIN 10  =  ErrorfromM2                  ;   /* M212 error */
        PIN 11  = !ErrorfromMOD0                ;   /* MOD 0 error */

/* oe */
        PIN 13  =  SelectM2     ; /* external 0v jumper onto 22v10 */

/* INPUTS/OUTPUTS */
        PIN 14  =  ResettoM2                    ;
        PIN 15  = !readInt                      ;   /* ic1 p13 */
        PIN 16  =  D0                           ;
        PIN 17  =  AnalysetoMOD0                ;    /*  d    */
        PIN 18  =  AnalysetoM2                  ;    /*  d    */
        PIN 19  = !notStatWR                    ;    /*  d    */
        PIN 20  =  ResettoMOD0                  ;    /*  d    */
        PIN 21  = !writeDMA                     ;    /* ic3 p10 */
        PIN 22  = !readDMA                      ;    /* ic3 p9  */
        PIN 23  = !writeInt                     ;    /* ic1 p11 */


notStatWR.d     = notIOW                        ;
notStatWR.ar = 'b'0 ; notStatWR.sp = 'b'0 ;

FIELD Register = [A1..A0]                       ;

                                              /*  Read & Write Declaratio
readsysM2       = notIOR & notSYSM2             ;
readErrM2       = readsysM2 & Register:[0]      ;
writesysM2      = notIOW & notSYSM2             ;
writeResetM2    = writesysM2 & Register:[0]     ;
writeAnalyseM2  = writesysM2 & Register:[1]     ;

readsysMOD0     = notIOR & notSYSMOD0           ;
readErrMOD0     = readsysMOD0 & Register:[0]    ;
writesysMOD0    = notIOW & notSYSMOD0           ;
writeResetMOD0  = writesysMOD0 & Register:[0]   ;
writeAnalyseMOD0= writesysMOD0 & Register:[1]   ;

readSys         = notIOR & notSYS               ;
writeSys        = notIOW & notSYS               ;
```

```
readErr          = readSys & Register:[0]          ;
writeDMA         = writeSys & Register:[2]         ;
readDMA          = readSys & Register:[2]          ;
writeInt         = writeSys & Register:[3]         ;
readInt          = readSys & Register:[3]          ;

                                                  /* Error flag to IBM            */
D0      =     !ErrorfromM2 & readErrM2
          £ !ErrorfromMOD0 & readErrMOD0          ;    /* IBM reads inverted
D0.oe   = readErr                                 ;

                                                  /* Latch reset & analyse from IBM
ResettoM2.d       =  D0 & writeResetM2
                   £ ResettoM2 & !writeResetM2
                   £ D0 & ResettoM2               ;
ResettoM2.oe      =  SelectM2                     ;
AnalysetoM2.d     =  D0 & writeAnalyseM2
                   £ AnalysetoM2 & !writeAnalyseM2
                   £ D0 & AnalysetoM2             ;
AnalysetoM2.oe    =  SelectM2                     ; /* active hi=M2 */
ResettoMOD0.d     =  D0 & writeResetMOD0
                   £ ResettoMOD0 & !writeResetMOD0
                   £ D0 & ResettoMOD0             ;

AnalysetoMOD0.d   =  D0 & writeAnalyseMOD0
                   £ AnalysetoMOD0 & !writeAnalyseMOD0
                   £ D0 & AnalysetoMOD0           ;

ResettoM2.ar    = 'b'0 ; ResettoM2.sp    = 'b'0 ;
AnalysetoM2.ar  = 'b'0 ; AnalysetoM2.sp  = 'b'0 ;
ResettoMOD0.ar  = 'b'0 ; ResettoMOD0.sp  = 'b'0 ;
AnalysetoMOD0.ar = 'b'0 ; AnalysetoMOD0.sp = 'b'0 ;
```

```
NAME            IC3v            ;
REVISION        01              ;
DATE            21/12/87        ;
DESIGNER        ARG             ;
COMPANY         QTM (Design) Ltd;
ASSEMBLY        m200            ;
DEVICE          p22v10          ;


/* INPUTS */
        PIN 1   =  clk_endDMA     ;
        PIN 2   =  A0             ;
        PIN 3   =  A1             ;
        PIN 4   = !DACK           ;
        PIN 5   =  InIntM2LADP    ;
        PIN 6   =  OutIntM2LADP   ;
        PIN 7   =  InIntMOD0LADP  ;
        PIN 8   =  OutIntMOD0LADP ;
        PIN 9   = !readDMA        ;
        PIN 10  = !writeDMA       ;
        PIN 11  = !SYSM2          ;
        PIN 13  =  resetDRV       ;

/* OUTPUTS */
        PIN 14  =  DRQ            ;       /* O              */
        PIN 15  =  TC             ;       /* I              */
        PIN 16  =  D0             ;       /* I/O            */
        PIN 17  =  D1             ;       /* O              */
        PIN 18  = !EndDMA         ;       /* I/O ic1 p10    */
        PIN 19  =  dirnofDMA      ;       /* I/O nc         */
        PIN 20  = !DMAactive      ;       /* I/O ic4 p18  actually true */
        PIN 21  =  DMAchM2notMOD0 ;       /* I/O ic4 p19    */
        PIN 22  =  DA1            ;       /* O              */
        PIN 23  =  DA0            ;       /* O              */


dirnofDMA       =  D0 & writeDMA
                 £ dirnofDMA & !writeDMA
                 £ D0 & dirnofDMA                   ; /* set dirn of txfer

D0              =  dirnofDMA                        ; /* echo dirn
D0.oe           =  readDMA                          ;

DMAchM2notMOD0  =  SYSM2 & writeDMA
                 £ DMAchM2notMOD0 & !writeDMA
                 £ SYSM2 & DMAchM2notMOD0           ; /* from SYS set chan

D1              = DMAchM2notMOD0                    ; /* echo selected chan
D1.oe           = readDMA                           ;

DA0             = A0 & !DACK £ !dirnofDMA & DACK    ;
DA1             = A1 & !DACK                        ;


/* Set DMA active when write dma is on , clr @ end of dma cycle £ reset */

EndDMA          =  TC & DACK
                 £ resetDRV     ;
DMAactive.d     ='b'1           ;  /* clocks an inactive o/s pal    */
```

```
DMAactive.ar     =  writeDMA     ;  /* this resets to active o/s pal */
DMAactive.sp     ='b'0           ;

DRQ      =    InIntM2LADP     &  DMAchM2notMOD0 &  dirnofDMA
           £  InIntMOD0LADP   & !DMAchM2notMOD0 &  dirnofDMA
           £  OutIntM2LADP    &  DMAchM2notMOD0 & !dirnofDMA
           £  OutIntMOD0LADP  & !DMAchM2notMOD0 & !dirnofDMA   ;
                                                    /* generate DMA req */
DRQ.oe   = !DMAactive                    ;    /* i/s pal inverted */
```

```
NAME            IC4v            ;
REVISION        01              ;
DATE            21/12/87        ;
DESIGNER        ARG             ;
COMPANY         QTM (Design) Ltd;
ASSEMBLY        m200            ;
DEVICE          p22v10          ;


/* INPUTS */
        PIN 1   = !StatWR       ;
        PIN 2   =  A9           ;
        PIN 3   =  A8           ;
        PIN 4   =  A7           ;
        PIN 5   =  A6           ;
        PIN 6   =  A4           ;
        PIN 7   =  AEN          ;
        PIN 8   = !IOW          ;
        PIN 9   =           nc1 ;
        PIN 10  =  seladdMOD0   ;
        PIN 11  =  seladdM2     ;
        PIN 13  = !DACK         ;
        PIN 14  =           nc2 ;
        PIN 18  =  DMAactive    ;
        PIN 19  =  DMAchM2notMOD0;
        PIN 20  =  A5           ;
        PIN 21  = !IOR          ;

/* OUTPUTS */
        PIN 15  = !SYSM2        ;
        PIN 16  = !SYSMOD0      ;
        PIN 17  = !SYS          ;
        PIN 22  = !CSMOD0LADP   ;
        PIN 23  = !CSM2LADP     ;


FIELD IBMaddr = [A9..A4]                    ;

LADPaddMOD0a =   IBMaddr:[150]  ; SYSaddMOD0a =  IBMaddr:[160]  ;
LADPaddMOD0b =   IBMaddr:[200]  ; SYSaddMOD0b =  IBMaddr:[210]  ;
LADPaddM2a   =   IBMaddr:[300]  ; SYSaddM2a   =  IBMaddr:[310]  ;
LADPaddM2b   =   IBMaddr:[250]  ; SYSaddM2b   =  IBMaddr:[260]  ;

 WE          = StatWR & IOW                         ; /*  short T3/T4 strobe

 LADPM2      =  LADPaddM2a & !AEN & seladdM2
              £ LADPaddM2b & !AEN & !seladdM2
              £ DACK & DMAchM2notMOD0 & DMAactive   ; /*  Link in IBM I/O ma
 LADPMOD0    =  LADPaddMOD0a & !AEN & seladdMOD0
              £ LADPaddMOD0b & !AEN & !seladdMOD0
              £ DACK & !DMAchM2notMOD0 & DMAactive  ; /*  Link In IBM I/O ma

 CSM2LADP    =   LADPM2 & ( WE £ IOR )              ;
 CSMOD0LADP  =   LADPMOD0 & ( WE £ IOR )            ;

 SYSM2       =   SYSaddM2a & !AEN & seladdM2
              £ SYSaddM2b & !AEN & !seladdM2        ; /* SubSys in IBM I/O m
 SYSMOD0     =   SYSaddMOD0a & !AEN & seladdMOD0
              £ SYSaddMOD0b & !AEN & !seladdMOD0    ; /* SubSys in IBM I/O m
```

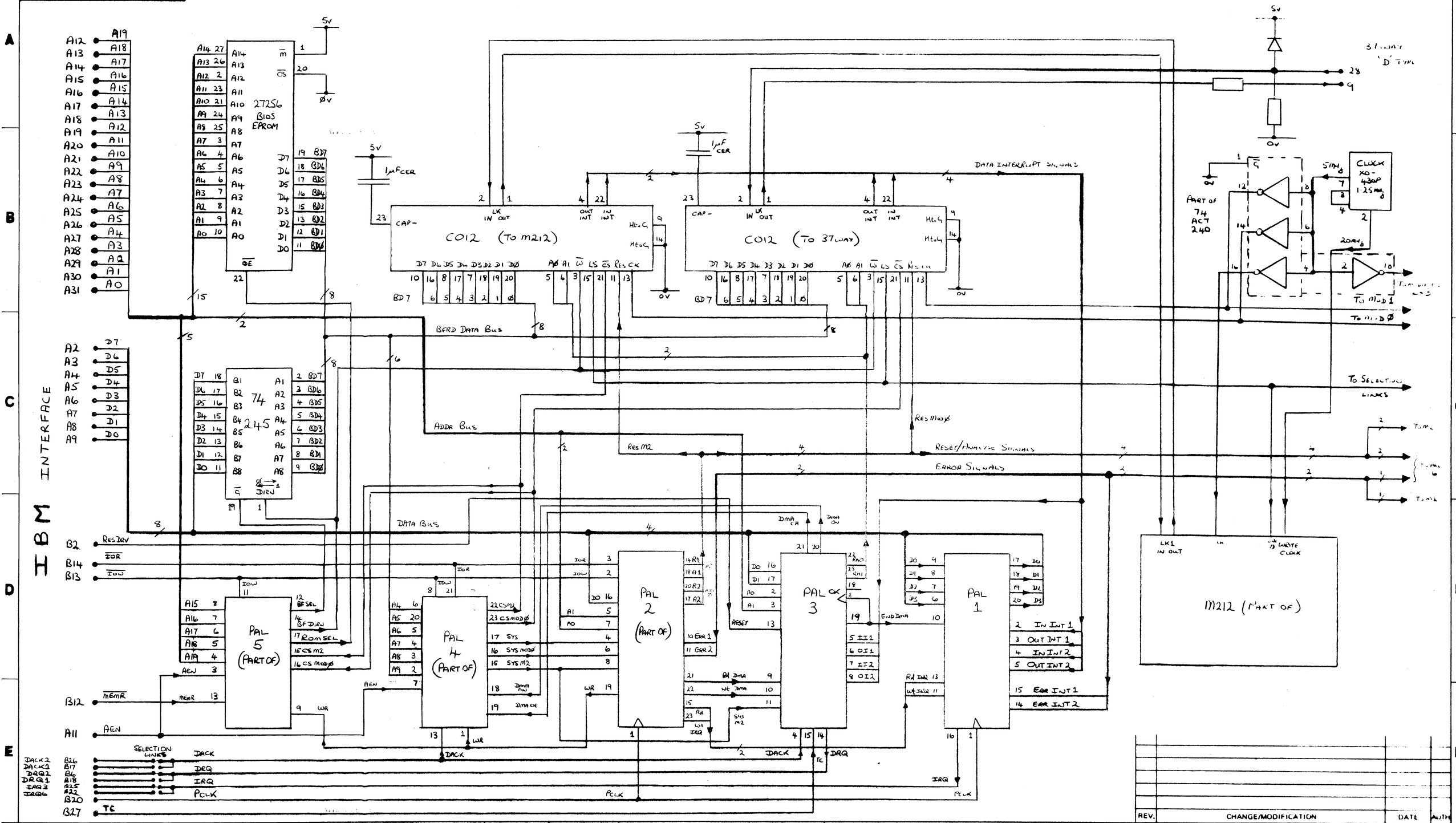
```
SYS          =    SYSM2 £ SYSMOD0                    ; /* BOARD sys in I/O ma
```

```
NAME            IC5v            ;
REVISION        01              ;
DATE            21/12/87        ;
DESIGNER        ARG             ;
COMPANY         QTM (Design) Ltd;
ASSEMBLY        m200            ;
DEVICE          P22v10          ;
/* CLOCK */
        PIN 1   =  ProcClockOut ;
        PIN 2   =  GND          ;       /* tied to 0V */
/* INPUTS */
        PIN 3   = !M2CS         ;
        PIN 4   =  M2A15        ;
        PIN 5   =  AEN          ;
        PIN 6   =  A19          ;
        PIN 7   =  A18          ;
        PIN 8   =  A17          ;
        PIN 9   =  A16          ;
        PIN 10  =  A15          ;
        PIN 11  = !StatWR       ;

        PIN 13  = !IOW          ;
        PIN 15  = !MEMR         ;
        PIN 17  = !CSMOD0LADP   ;
        PIN 18  = !CSM2LADP     ;

/* OUTPUTS */
        PIN 14  = !BfrSel       ;
        PIN 16  = !BfrDirn      ;
        PIN 19  = !ROMSEL       ;
        PIN 20  = !M2ramlo      ;
        PIN 21  = !M2ramhi      ;

        PIN 22  =  VCC          ;  /* Tied to VCC */
        PIN 23  = !Wait         ;

FIELD IBMbios = [A19..A15]                      ;


 ROMSEL  = IBMbios:[D0000] & !AEN & MEMR        ; /* Define ROM in IBM Mem Ma
                                                  /* as a read only periphera
 M2ramlo = M2CS & !M2A15                        ;
 M2ramhi = M2CS & M2A15                         ;

 BfrSel  = CSM2LADP £ CSMOD0LADP £ ROMSEL       ;
 BfrDirn = StatWR £ IOW                         ; /* a wide pulse */

Wait.d   = M2CS                                 ; /* delay by M2 clock */
Wait.sp  = 'b'0 ; Wait.ar  = 'b'0 ; Wait.oe = 'b'1 ;
```

```
NAME            IC6v            ;
REVISION        01              ;
DATE            05/01/88        ;
DESIGNER        ARG             ;
COMPANY         QTM (Design) Ltd;
ASSEMBLY        m200            ;
DEVICE          p22v10          ;


/* INPUTS */
        PIN 1   =           nc1             ;
        PIN 2   =  fromIBM                  ;
        PIN 3   = !fromMOD0SS               ;
        PIN 4   =  AnalysefromIBM           ;   /* active hi   */
        PIN 5   =  ResetfromIBM             ;   /* active hi   */
        PIN 6   = !ErrorfromSubSys          ;   /* active   lo */
        PIN 7   = !AnalysefromUP            ;   /* active   lo */
        PIN 8   = !ResetfromUP              ;   /* active   lo */
        PIN 9   = !ErrorfromDOWNorMOD0      ;   /* active   lo */
        PIN 10  =  AnalysefromMOD0SS        ;   /* active hi   */
        PIN 11  =  ResetfromMOD0SS          ;   /* active hi   */
        PIN 13  = !ErrorfromModules         ;   /* active   lo */

/* OUTPUTS */
        PIN 14  = !ErrortoMOD0SS            ;   /* active   lo */
        PIN 15  =  AnalysetoModules         ;   /* active hi   */
        PIN 16  =  ResettoModules           ;   /* active hi   */
        PIN 17  = !ErrortoUPandIBM          ;   /* active   lo */
        PIN 18  =  ResettoMOD0              ;   /* active hi   */
        PIN 19  =  AnalysetoMOD0            ;   /* active hi   */
        PIN 20  = !AnalysetoDOWN            ;   /* active   lo */
        PIN 21  = !ResettoDOWN              ;   /* active   lo */
        PIN 22  = !AnalysetoSubSys          ;   /* active   lo */
        PIN 23  = !ResettoSubSys            ;   /* active   lo */


ResettoMOD0       =   ResetfromIBM    &  fromIBM
                  £   ResetfromUP     & !fromIBM              ;
AnalysetoMOD0     =   AnalysefromIBM  &  fromIBM
                  £   AnalysefromUP   & !fromIBM              ;

ResettoDOWN       =   ResettoMOD0                           ;
AnalysetoDOWN     =   AnalysetoMOD0                         ;

ResettoModules    =   ResettoMOD0       & !fromMOD0SS
                  £ ResetfromMOD0SS     &  fromMOD0SS       ;
AnalysetoModules  =   AnalysetoMOD0     & !fromMOD0SS
                  £ AnalysefromMOD0SS   &  fromMOD0SS       ;

ResettoSubSys     =   ResettoModules                        ;
AnalysetoSubSys   =   AnalysetoModules                      ;

ErrortoUPandIBM   =   ErrorfromDOWNorMOD0                   ;
ErrortoMOD0SS     =   ErrorfromModules
                  £ ErrorfromSubSys                         ;
```

7 Logic Diagram

DRAWING No.



REV.	CHANGE/MODIFICATION	DATE	AUTH

SPECIAL NOTICES

THE FOLLOWING DOCUMENTS SHOULD BE READ IN CONJUNCTION WITH THIS DRAWING :

		DATE
DRAWN		10/JAN/88
ENG. APPR.		
AUTHORISED		

UNLESS OTHERWISE STATED THE INFORMATION CONTAINED WITHIN THIS DRAWING IS IN COMPLIANCE WITH THE UNDERMENTIONED STANDARDS MANUALS.

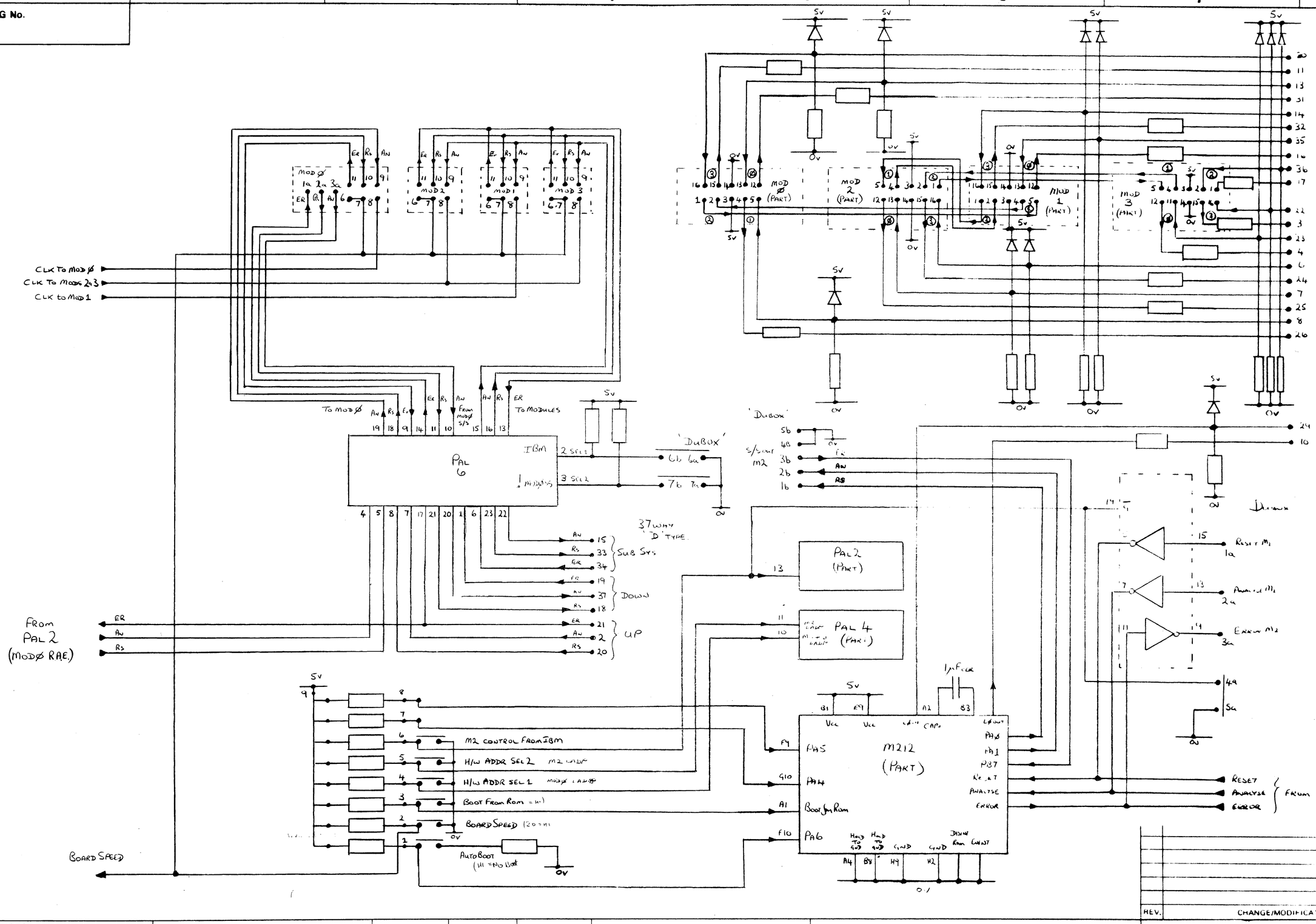
TITLE M2000 IBM INTERFACE

SHEET 1 OF 3

CONTRACT

DRAWING No.

DRAWING No.



REV.	CHANGE/MODIFICATION	DATE	APP.

SPECIAL NOTICES

THE FOLLOWING DOCUMENTS SHOULD BE READ IN CONJUNCTION WITH THIS DRAWING:

		DATE
DRAWN		10 JAN 85
ENG. APPR.		
AUTHORISED		

UNLESS OTHERWISE STATED THE INFORMATION CONTAINED WITHIN THIS DRAWING IS IN COMPLIANCE WITH THE UNDERMENTIONED STANDARDS MANUALS.

TITLE LINK & RESET CIRCUITS SHEET 2 of 3

CONTRACT

DRAWING No.

DRAWING No.

SEE SHEET 2

PAL5 (PART) 22V10

M212

ST506 34WAY FOR A SINGLE DISK

ST506 20WAY FOR A SINGLE DISK

75179

74F373

6225 32K x 8 -10

100R

22K

330

10nF

680R

300R

15

150R

6n8

3n3

30

1K

Vcc

0V

WAIT STATE TIMING TO ADD A SINGLE WAIT STATE

REV.	CHANGE/MODIFICATION	DATE

SPECIAL NOTICES

EVENT I/P & MODEM REG ?

THE FOLLOWING DOCUMENTS SHOULD BE READ IN CONJUNCTION WITH THIS DRAWING :

		DATE
DRAWN		11 JAN 86
ENG. APPR.		
AUTHORISED		

UNLESS OTHERWISE STATED THE INFORMATION CONTAINED WITHIN THIS DRAWING IS IN COMPLIANCE WITH THE UNDERMENTIONED STANDARDS MANUALS.

TITLE

M212 INTERFACE

CONTRACT

DRAWING No.