Микропроцессорный комплект  
К1811

Техническое описание

Версия 1.0 (January 15, 2022)

# Введение

Данный документ описывает микросхемы микропроцессорного комплекта серии 1811.

|  |  |  |
| --- | --- | --- |
| Микросхема | Оригинал DEC | Описание |
| КН1811ВМ1 | DC302E 21-15541-AB | центральный процессорный элемент |
| КН1811ВУ1 | DC303E 23-001C7-AA | управляющий элемент, MicROM + PLA, CS=0 |
| КН1811ВУ2 | DC303E 23-203C7-AA | управляющий элемент, MicROM + PLA, CS=2 |
| КН1811ВУ3 | DC303E 23-002C7-AA | управляющий элемент, MicROM + PLA, CS=1 |
| КН1811ВТ1 | DC304E 21-15542-01 | диспетчер памяти |

# Шина микроинструкций MIB

При низком тактовом сигнале по шине передается код микроинструкции от управляющего элемента в процессорный. При высоком уровне тактового сигнала по шине микроинструкций передается управляющее слово. Сигналы на шине микроинструкций **НЕ** инвертированы.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Разряд | Имя | Описание | | |
| **M[15]** | **nMME** | MMU разрешен, низкий активный уровень сообщает логике процессорной платы что должен быть выполнен микроцикл трансляции адреса. Линия активируется микросхемой MMU или логикой адресации ODT. | | |
|  | **INIT** | Сброс периферийных устройств. Активный высокий, вызывает генерацию активного сигнала **nBINIT** на внешней шине | | |
| **M[13]** | **IAKO** | Высокий уровень подтверждает прерывание, на линии **nBIAKO** внешней шины появляется активный низкий уровень | | |
| **M[12,9,8]** | **AIO** | Address Input-Output кодирование | | |
| **000** | **AWO** | Выдача адреса, далее следует запись данных |
| **001** | **ARW** | Выдача адреса, далее следует чтение-модификация-запись данных |
| **010** | **-** | Не используется |
| **011** | **ARO** | Выдача адреса, далее следует чтение данных |
| **100** | **DOUTB** | Выдача данных, байтовая запись |
| **101** | **DOUT** | Выдача данных, словная запись |
| **110** | **DIN** | Прием данных |
| **111** | **NOP** | Нет операции шины |
| **M[11]** | **nBR** | Процессорный элемент сообщает следует ли выполнить ветвление микропрограммы в зависимости от закодированного микроинструкцией условия и флагов PSW/ALU. | | |
| **M[10]** | **nTEST** | Тестирование Control Chip, выбранная микросхема DC302/1811ВУх в следующем микроцикле выдает на шину микроинструкций NAF (Next Address Field) вместо кода микроинструкции. На плате процессора в рабочем режиме никогда не активируется, используется только для стендового тестирования | | |
| **M[7]** | **nSYNC** | Низкий уровень означает наличие активного цикла обмена по шине QBus | | |
| **M[6,5,4]** | **CTRL** | Управление операцией Control Chip | | |
| **000** | load PLM register address from return register (affects MC/NA/AXT) | |
| **001** | load PLM register or MMU control register | |
| **010** | set stack overflow in Control Chip | |
| **011** | enable address conversion mode in Control Chip | |
| **100** | transfer priority and T-bit to Control Chip | |
| **101** | transfer priority to Control Chip | |
| **110** | transfer T-bit to Control Chip | |
| **111** | Нет операции | |
| **M[3,2,1,0]** | **GPO** | Линии управления общего назначения | | |
| **1xxx** | Нет активных сигналов | |
| **0001** | SRUN – выборка и дешифрация инструкции PDP-11 | |
| **0101** | Очистка запроса прерывания от таймера (EVNT) | |
| **0110** | Очистка запроса прерывания по сбою питания (ACLO) | |
| **0111** | Загрузка регистра расширения адреса ODT | |

# Внутренняя шина данных AD

При высоком уровне тактового сигнала по шине передаются данные адреса, чтения-записи внешней шины, а также данные из регистра быстрого чтения. При низком тактовом сигнале по шине сервисное слово содержащее информацию о состоянии процессора и запросах на прерывание. Сигналы на внутренней шине данных **НЕ** инвертированы

|  |  |  |  |
| --- | --- | --- | --- |
| Разряд | Имя | Описание | Приоритет |
| **AD[15]** | **WAIT** | Данный сигнал формируется внутри управляющего элемента ВУх и не виден на внешних линиях. | 16 |
| **AD[14]** | **TBIT** | Данный сигнал формируется внутри управляющего элемента ВУх и не виден на внешних линиях. Содержит копию разряда 4 слова состояния процессора, записанную микропрограммой из процессорного элемента. | 7 |
| **AD[13]** | **YSTK** | Признак переполнения стека (SP < 4008). Данный сигнал формируется внутри управляющего элемента ВУх и не виден на внешних линиях. Записывается процессорным элементом при выполнении операций со стеком. | 8 |
| **AD[12]** | **EVNT** | Запрос прерывания от таймера (вектор 1008), высокий уровень | 11 |
| **AD[11]** | **IRQ4** | Векторное прерывание от внешних устройств, линия IRQ4L, активный уровень высокий | 14 |
| **AD[10]** | **IRQ5** | Векторное прерывание от внешних устройств, линия IRQ5L активный уровень высокий | 13 |
| **AD[9]** | **IRQ6** | Векторное прерывание от внешних устройств, линия IRQ6L активный уровень высокий | 12 |
| **AD[8]** | **IRQ7** | Векторное прерывание от внешних устройств, линия IRQ7L активный уровень высокий | 10 |
| **AD[7]** | **nACLO** | Пропадание питания, активный уровень низкий (при нормальной работе на этом разряде присутствует высокий уровень) | 9 |
| **AD[6]** | **-** | Всегда читается низкий уровень | - |
| **AD[5]** | **HALT** | Запрос на останов, активный уровень высокий | 15 |
| **AD[4]** | **ECTL** | Ошибка управляющего элемента (отсутствует активный CS) | 2 |
| **AD[3]** | **ABRT** | Прерывание операции со стороны MMU | 3 |
| **AD[2]** | **PERR** | Ошибка четности шины | 5 |
| **AD[1]** | **BUSTO** | Тайм-аут шины | 4 |
| **AD[0]** | **DCLO** | Пропадание постоянного питания (высокий уровень активный) | 1 |

# Регистр быстрого ввода

Регистр быстрого ввода читается микропрограммой по упрощенной процедуре без генерации полноценного цикла внешней шины.

|  |  |  |
| --- | --- | --- |
| Разряд | Имя | Описание |
| **FDIN[15-9]** | **START** | Старшие биты 16-разрядного стартового адреса после начального сброса (младшие разряды адреса назначаются нулевыми). |
| **FDIN[8]** | **A173** | 0 – стартовать по адресу заданному AD[15-9] 1- стартовать по фиксированному адресу 1730008 |
| **FDIN[7]** | **DPOK** | Признак начального старта после сброса |
| **FDIN[2]** | **HODT** | 0 – инструкция HALT вызывает ODT 1 - инструкция HALT вызывает исключение 108 |
| **FDIN[1]** | **BOOT** | Выбор способа начального старта 00 - старт по содержимому вектора 248 01 – переход в ODT 10 – переход по 173008 или FDIN[15-9] 11 – старт из расширенного MicROM |
| **FDIN[0]** |  |

# Система микрокоманд К1811

|  |  |  |
| --- | --- | --- |
| ccccc | chip5 | Control chip Select – 0..31, for jmp instruction only |
| dddddd | addr6 | Entry address – 0..63, for jmp instruction only |
| mmmm | mode | Extra control bits |
| S | S suffix | 0 – saves ALU flags into PSW. ALU flags valid till next microinstruction, PSW flags are permanent |
| iiii iiii | data8 | Unsigned 8-bit value {m[7:4], m[3:0]), for AL is sign extended |
| aaaa | Ra | Destination register index, 12 – PDP source, 13 – PDP destination, 8 – PSW, R6 is mode dependent |
| bbbb | Rb | Source register index, 12 – PDP source, 13 – PDP destination, 8 - PSW, R6 mode dependent |

|  |  |  |  |
| --- | --- | --- | --- |
| 0000 0ccc ccdd dddd | JMP | chip5, addr6 | select control by chip5 field , LC = addr6; |
| 0000 1fff dddd dddd | Conditional jumps within 256 microinstructions page | | |
| 0000 1000 dddd dddd | JNA | addr8 | if (ALU\_N) LC[7:0] = addr8; |
| 0000 1001 dddd dddd | JZA | addr8 | if (ALU\_ZL & ALU\_ZH) LC[7:0] = addr8; |
| 0000 1010 dddd dddd | JCA | addr8 | if (ALU\_C) LC[7:0] = addr8; |
| 0000 1011 dddd dddd | JVA | addr8 | if (ALU\_V) LC[7:0] = addr8; |
| 0000 1100 dddd dddd | JN | addr8 | if (PSW[3]) LC[7:0] = addr8; |
| 0000 1101 dddd dddd | JZ | addr8 | if (PSW[2]) LC[7:0] = addr8; |
| 0000 1110 dddd dddd | JC | addr8 | if (PSW[0]) LC[7:0] = addr8; |
| 0000 1111 dddd dddd | JM | addr8 | if (PSW[15:14] != 00) LC[7:0] = addr8; |
| 0001 xxxw xxxx aaaa | Single operand commands | | |
| 0001 0000 mmmm aaaa | oq | areg | data output, ends qbus cycle, 101 nanocmd for PSW address |
| 0001 0001 mmmm aaaa | oir | mmmm, areg | IR STB, mi[7:4] -> control word [3:0], 001 nanocmd |
| 0001 0010 mmmm aaaa | ops | mmmm, areg | Set PRI + T, 100 nanocmd, M[7] – do external INIT |
| 0001 0011 mmmm aaaa | oplm | mmmm, areg | Set PLM register, 000 nanocmd |
| 0001 0100 mmmm aaaa | RWI | mmmm, areg | address Ra, read word, Ra = Ra + 2 |
| 0001 0101 mmmm aaaa | RBI | mmmm, areg | address Ra, read byte, Ra = Ra + 1 |
| 0001 0110 mmmm aaaa | RW | mmmm, areg | address Ra, read word |
| 0001 0111 mmmm aaaa | RB | mmmm, areg | address Ra, read byte |
| 0001 1000 mmmm aaaa | WWI | mmmm, areg | address Ra, write word, Ra = Ra + 2 |
| 0001 1001 mmmm aaaa | WBI | mmmm, areg | address Ra, write byte, Ra = Ra + 1 |
| 0001 1010 mmmm aaaa | WW | mmmm, areg | address Ra, write word |
| 0001 1011 mmmm aaaa | WB | mmmm, areg | address Ra, write byte |
| 0001 1100 mmmm aaaa | RWWI | mmmm, areg | address Ra, read-modify-write word, Ra = Ra + 2 |
| 0001 1101 mmmm aaaa | RWBI | mmmm, areg | address Ra, read-modify-write byte, Ra = Ra + 1 |
| 0001 1110 mmmm aaaa | RWW | mmmm, areg | address Ra, read-modify-write word |
| 0001 1111 mmmm aaaa | RWB | mmmm, areg | address Ra, read-modify-write byte |
| 0xxx iiii iiii aaaa | Literal commands (immediate 8 bit signed/unsigned value) | | |
| 0010 iiii iiii aaaa | LL | udata8, areg | Ra = udata8 |
| 0011 iiii iiii aaaa | CL | udata8, areg | udata8 - Ra |
| 0100 iiii iiii aaaa | AL | data8, areg | Ra = Ra + sext(data8) |
| 0101 iiii iiii aaaa | OL | udata8, areg | Ra = Ra | udata8 |
| 0110 iiii iiii aaaa | NL | udata8, areg | Ra = Ra & udata8 |
| 0111 iiii iiii aaaa | TL | udata8, areg | Ra & udata8 |
| 1xxx xxsw bbbb aaaa | Two operand commands | | |
| 1000 00s0 bbbb aaaa | SRWs | breg, areg | Ra = Rb >> 1 | Rb[15]<<15 |
| 1000 00s1 bbbb aaaa | SRBs | breg, areg | Ra[7:0] = Rb[7:0]>> 1 | Rb[7]<<7 |
| 1000 01s0 bbbb aaaa | SRWCs | breg, areg | Ra = Rb >> 1 | C<<15 |
| 1000 01s1 bbbb aaaa | SRBCs | breg, areg | Ra[7:0] = Rb[7:0]>> 1 | C << 7 |
| 1000 10s0 bbbb aaaa | SLWs | breg, areg | Ra = Rb << 1 |
| 1000 10s1 bbbb aaaa | SLBs | breg, areg | Ra[7:0] = Rb[7:0]<<1 |
| 1000 11s0 bbbb aaaa | SLWCs | breg, areg | Ra = Rb << 1 | C |
| 1000 11s1 bbbb aaaa | SLBCs | breg, areg | Ra[7:0] = Rb[7:0]<< 1 | C |
| 1001 00s0 bbbb aaaa | TCWs | breg, areg | Ra = ~Rb + 1 |
| 1001 00s1 bbbb aaaa | TCBs | breg, areg | Ra[7:0] = ~Rb[7:0] +1 |
| 1001 01s0 bbbb aaaa | TCWCs | breg, areg | Ra = ~Rb + ~C |
| 1001 01s1 bbbb aaaa | TCBCs | breg, areg | Ra[7:0] = ~Rb[7:0] + ~C |
| 1001 10s0 bbbb aaaa | CWs | breg, areg | Rb - Ra, compare words |
| 1001 10s1 bbbb aaaa | CBs | breg, areg | Rb[7:0]- Ra[7:0], compare byte |
| 1001 11s0 bbbb aaaa | OCWs | breg, areg | Ra = ~Rb |
| 1001 11s1 bbbb aaaa | OCBs | breg, areg | Ra[7:0]= ~Rb[7:0] |
| 1010 00s0 bbbb aaaa | AWs | breg, areg | Ra = Ra + Rb |
| 1010 00s1 bbbb aaaa | ABs | breg, areg | Ra[7:0]= Ra[7:0] + Rb[7:0] |
| 1010 01s0 bbbb aaaa | AWCs | breg, areg | Ra = Ra + Rb+ C |
| 1010 01s1 bbbb aaaa | ABCs | breg, areg | Ra[7:0]= Ra[7:0] + Rb[7:0] + C |
| 1010 10s0 bbbb aaaa | ICWs | breg, areg | Ra = Rb + 1 |
| 1010 10s1 bbbb aaaa | ICBs | breg, areg | Ra[7:0]= Rb[7:0] + 1 |
| 1010 11s0 bbbb aaaa | ACWs | breg, areg | Ra = Rb + C |
| 1010 11s1 bbbb aaaa | ACBs | breg, areg | Ra[7:0]= Rb[7:0] + C |
| 1011 00s0 bbbb aaaa | SWs | breg, areg | Ra = Ra - Rb |
| 1011 00s1 bbbb aaaa | SBs | breg, areg | Ra[7:0]= Ra[7:0] - Rb[7:0] |
| 1011 01s0 bbbb aaaa | SWCs | breg, areg | Ra = Ra - Rb- C |
| 1011 01s1 bbbb aaaa | SBCs | breg, areg | Ra[7:0]= Ra[7:0] - Rb[7:0] - C |
| 1011 10s0 bbbb aaaa | DCWs | breg, areg | Ra = Rb - 1 |
| 1011 10s1 bbbb aaaa | DCBs | breg, areg | Ra[7:0]= Rb[7:0] - 1 |
| 1011 11s0 bbbb aaaa | SCWs | breg, areg | Ra = Rb - C |
| 1011 11s1 bbbb aaaa | SCBs | breg, areg | Ra[7:0]= Rb[7:0] - C |
| 1100 00s0 bbbb aaaa | MWs | breg, areg | Ra = Rb |
| 1100 00s1 bbbb aaaa | MBs | breg, areg | Ra[7:0]= Rb[7:0] |
| 1100 01s0 bbbb aaaa | SXWs | breg, areg | Ra = N ? 0xFFFF : 0 |
| 1100 01s1 bbbb aaaa | SXBs | breg, areg | Ra[7:0]= N ? 0xFF : 0 |
| 1100 10s0 bbbb aaaa | ZWs | Areg | Ra = 0 |
| 1100 10s1 bbbb aaaa | ZBs | Areg | Ra[7:0]= 0 |
| 1100 11s0 bbbb aaaa | BDWs | breg, areg | Binary-decimal correction after subtraction |
| 1100 11s1 bbbb aaaa | BDBs | breg, areg | Binary-decimal correction after subtraction |
| 1101 0000 mmmm aaaa | INW | mmmm. areg | data input, swap on odd addr |
| 1101 0001 mmmm aaaa | INB | mmmm. areg | data input |
| 1101 0010 mmmm aaaa | INWQ | mmmm. areg | data input, swap on odd addr, end bus cycle |
| 1101 0011 mmmm aaaa | INBQ | mmmm. areg | data input, end bus cycle |
| 1101 0100 mmmm aaaa | IRW | mmmm. areg | IR input, swap on odd addr, mi[7:4] -> control[3:0] |
| 1101 0101 mmmm aaaa | IRB | mmmm. areg | IR input, mi[7:4] -> control[3:0] |
| 1101 0110 mmmm aaaa | IRWQ | mmmm. areg | IR input, swap on odd addr, mi[7:4] -> control[3:0] , end bus cycle |
| 1101 0111 mmmm aaaa | IRBQ | mmmm. areg | IR input, mi[7:4] -> control[3:0] , end bus cycle |
| 1101 1000 mmmm aaaa | ISW | mmmm. areg | PSW input, swap on odd addr |
| 1101 1001 mmmm aaaa | ISB | mmmm. areg | PSW input |
| 1101 1010 mmmm aaaa | ISWQ | mmmm. areg | PSW input, swap on odd addr, end bus cycle |
| 1101 1011 mmmm aaaa | ISBQ | mmmm. areg | PSW input, end bus cycle |
| 1101 1100 mmmm aaaa | IVW | mmmm. areg | IAKO, swap on odd addr |
| 1101 1101 mmmm aaaa | IVB | mmmm. areg | IAKO |
| 1101 1110 mmmm aaaa | IVWQ | mmmm. areg | IAKO, swap on odd addr, end bus cycle |
| 1101 1111 mmmm aaaa | IVBQ | mmmm. areg | IAKO, end bus cycle |
| 1110 00s0 bbbb aaaa | NCWs | breg, areg | Ra = Ra & ~Rb |
| 1110 00s1 bbbb aaaa | NCBs | breg, areg | Ra[7:0] = Ra[7:0] & ~Rb[7:0] |
| 1110 01s0 bbbb aaaa | ORWs | breg, areg | Ra = Ra | Rb |
| 1110 01s1 bbbb aaaa | ORBs | breg, areg | Ra[7:0] = Ra[7:0] | Rb[7:0] |
| 1110 10s0 bbbb aaaa | XOWs | breg, areg | Ra = Ra ^ Rb |
| 1110 10s1 bbbb aaaa | XOBs | breg, areg | Ra[7:0] = Ra[7:0] ^ Rb[7:0] |
| 1110 11s0 bbbb aaaa | XCHWs | breg, areg | Ra = {Rb[7:0], Rb[15:0]} (swap) |
| 1110 11s1 bbbb aaaa | XCHBs | breg, areg | Ra[7:0] = sign (Rb[7:0]) |
| 1111 00s0 bbbb aaaa | TZWs | breg, areg | Ra, set NZ, VC=0 |
| 1111 00s1 bbbb aaaa | TZBs | breg, areg | Ra[7:0] , set NZ, VC=0 |
| 1111 0100 bbbb aaaa | CAWS | breg, areg | if (C) {Ra = Ra + Rb} |
| 1111 0101 bbbb aaaa | CABS | breg, areg | if (C) {Ra[7:0] = Ra[7:0] + Rb[7:0]} |
| 1111 0110 bbbb aaaa | LXW | breg, areg | Ra = sext(Rb[7:0]) << 1 |
| 1111 0111 bbbb aaaa | LXB | breg, areg | Ra[7:0] = sext(Rb[7:0]) << 1 |
| 1111 10s0 bbbb aaaa | TWs | breg, areg | Ra & Rb |
| 1111 10s1 bbbb aaaa | TBs | breg, areg | Ra[7:0] & Rb[7:0] |
| 1111 1100 bbbb aaaa |  |  | Previous mode (PSW[13:12) register SP access |
| 1111 1101 bbbb aaaa |  |  | Previous mode (PSW[13:12) register SP access |
| 1111 1110 bbbb aaaa | CAWI | breg, areg | If (PSW cond) {Ra = Ra + Rb} (PDP-11 conditional branches) |
| 1110 1111 bbbb aaaa | NOP |  | No operation |

# Регистры MMU

|  |  |  |
| --- | --- | --- |
| Адрес | Имя | Описание |
| 177723008 | **KPDR0** | Kernel mode page descriptor register 0 |
| 177723028 | **KPDR1** | Kernel mode page descriptor register 1 |
| 177723048 | **KPDR2** | Kernel mode page descriptor register 2 |
| 177723068 | **KPDR3** | Kernel mode page descriptor register 3 |
| 177723108 | **KPDR4** | Kernel mode page descriptor register 4 |
| 177723128 | **KPDR5** | Kernel mode page descriptor register 5 |
| 177723148 | **KPDR6** | Kernel mode page descriptor register 6 |
| 177723168 | **KPDR7** | Kernel mode page descriptor register 7 |
| 177723408 | **KPAR0** | Kernel mode page address register 0 |
| 177723428 | **KPAR1** | Kernel mode page address register 1 |
| 177723448 | **KPAR2** | Kernel mode page address register 2 |
| 177723468 | **KPAR3** | Kernel mode page address register 3 |
| 177723508 | **KPAR4** | Kernel mode page address register 4 |
| 177723528 | **KPAR5** | Kernel mode page address register 5 |
| 177723548 | **KPAR6** | Kernel mode page address register 6 |
| 177723568 | **KPAR7** | Kernel mode page address register 7 |
| 177725168 | **SR3** | Status Register 3 |
| 177775728 | **SR0** | Status Register 0 |
| 177775748 | **SR1** | Status Register 1 |
| 177775768 | **SR2** | Status Register 2 |
| 177776008 | **KPDR0** | User mode page descriptor register 0 |
| 177776028 | **KPDR1** | User mode page descriptor register 1 |
| 177776048 | **KPDR2** | User mode page descriptor register 2 |
| 177776068 | **KPDR3** | User mode page descriptor register 3 |
| 177776108 | **KPDR4** | User mode page descriptor register 4 |
| 177776128 | **KPDR5** | User mode page descriptor register 5 |
| 177776148 | **KPDR6** | User mode page descriptor register 6 |
| 177776168 | **KPDR7** | User mode page descriptor register 7 |
| 177776408 | **KPAR0** | User mode page address register 0 |
| 177776428 | **KPAR1** | User mode page address register 1 |
| 177776448 | **KPAR2** | User mode page address register 2 |
| 177776468 | KPAR3 | User mode page address register 3 |
| 177776508 | KPAR4 | User mode page address register 4 |
| 177776528 | KPAR5 | User mode page address register 5 |
| 177776548 | KPAR6 | User mode page address register 6 |
| 177723568 | KPAR7 | Kernel mode page address register 7 |

# Разное

Ветвление микроадреса

Для управления ветвлением микропрограммы адресное поле микрокоманды может быть изменено следую­щими четырьмя способами:

- разряды [0:3] определенных адресов из­меняются логикой МА под управлением дан­ных из РД (используется только MiCROM CS0 для декодирования основных команд PDP-11);

- разряды [0:7] любого адреса заменяют данными из разрядов [0:7] РМ под управле­нием логики ВМ (условное ветвление);

- разряды [0:8] любого адреса генерируют­ся формирователем адреса МК в соответст­вии с данными в РД;

- разряды [0:8] адреса формируются в процессе выборки микрокоманды из програм­мируемой логической матрицы с помощью ло­гического сложения.