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/FPP control register | |
| **010** | set stack overflow in Control Chip | |
| **011** | address conversion User mode in MMU | |
| **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 ВУx (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 FDIN[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

## Поля аргументов микрокоманд

|  |  |  |
| --- | --- | --- |
| 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 |

## Таблица микрокоманд

|  |  |  |  |
| --- | --- | --- | --- |
| 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 | mmmm, 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 |
| 1101 0001 mmmm aaaa | INB | mmmm. areg | data input, swap on odd addr |
| 1101 0010 mmmm aaaa | INWQ | mmmm. areg | data input, end bus cycle |
| 1101 0011 mmmm aaaa | INBQ | mmmm. areg | data input, odd swap, end bus cycle |
| 1101 0100 mmmm aaaa | IRW | mmmm. areg | IR input, mi[7:4] -> control[3:0] |
| 1101 0101 mmmm aaaa | IRB | mmmm. areg | IR input, odd swap, mi[7:4] -> control[3:0] |
| 1101 0110 mmmm aaaa | IRWQ | mmmm. areg | IR input, mi[7:4] -> control[3:0] , end bus cycle |
| 1101 0111 mmmm aaaa | IRBQ | mmmm. areg | IR input, odd swap, mi[7:4] -> control[3:0] , end bus cycle |
| 1101 1000 mmmm aaaa | ISW | mmmm. areg | PSW input |
| 1101 1001 mmmm aaaa | ISB | mmmm. areg | PSW input, odd swap |
| 1101 1010 mmmm aaaa | ISWQ | mmmm. areg | PSW input, end bus cycle |
| 1101 1011 mmmm aaaa | ISBQ | mmmm. areg | PSW input, odd swap, end bus cycle |
| 1101 1100 mmmm aaaa | IVW | mmmm. areg | IAKO |
| 1101 1101 mmmm aaaa | IVB | mmmm. areg | IAKO, odd swap |
| 1101 1110 mmmm aaaa | IVWQ | mmmm. areg | IAKO, end bus cycle |
| 1101 1111 mmmm aaaa | IVBQ | mmmm. areg | IAKO, odd swap, 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 |

## Микрокоманды ввода-вывода

|  |  |  |  |
| --- | --- | --- | --- |
| Hex | Name | cmd/io | Description |
|  | Address output | | |
| 14ma | **RWI** | 111/011 | address Ra, read word, Ra = Ra + 2 |
| 15ma | **RBI** | 111/011 | address Ra, read byte, Ra = Ra + 1 |
| 16ma | **RW** | 111/011 | address Ra, read word |
| 17ma | **RB** | 111/011 | address Ra, read byte |
| 18ma | **WWI** | 111/000 | address Ra, write word, Ra = Ra + 2 |
| 19ma | **WBI** | 111/000 | address Ra, write byte, Ra = Ra + 1 |
| 1Ama | **WW** | 111/000 | address Ra, write word |
| 1Bma | **WB** | 111/000 | address Ra, write byte |
| 1Cma | **RWWI** | 111/001 | address Ra, read-modify-write word, Ra = Ra + 2 |
| 1Dma | **RWBI** | 111/001 | address Ra, read-modify-write byte, Ra = Ra + 1 |
| 1Ema | **RWW** | 111/001 | address Ra, read-modify-write word |
| 1Fma | **RWB** | 111/001 | address Ra, read-modify-write byte |
|  | | | M[4] – force kernel CPU mode address translation |
| M[5] – 1 – PDP-11 instruction fetch |
| M[6] – 1-current/ 0 - previous CPU mode address translation / disable ODT if M[7] is 1 |
| M[7]- disable address translation in MMU |
|  | Data output | | |
| 10ma | **OQ** | 1x1/10w | data output, ends qbus cycle, 101 nanocmd for PSW address only |
|  | | | M[4] – FPP register bank select, 0 – B, 1- A |
| M[5] – FPP register selector FA[1] |
| M[6] – toggle lower/upper half of 32-bit FPP register |
| M[7] – FPP register selector ~FA[3] |
|  | | | M[7:5] = x00 – no access M[7:5] = x01 –extra FPP register access (FS[16], FS[17]) M[7:5] = 010 – AC6 M[7:5] = 011 – AC7 M[7:5] = 110 – fsrc/fdst field from PDP instruction M[7:5] = 111 –ac field from PDP instruction |
| 11ma | **OIR** | 001/10w | IR STB, mi[7:4] -> control word [3:0] |
| 12ma | **OPS** | 100/10w | Set PRI + T |
|  | | | M[7] – do external INIT, MMU also tracks this bit |
| 13ma | **OPLM** | 000/10w | Set PLM register |
|  | Data input | | |
| D0ma | **INW** | 111/110 | input word |
| D1ma | **INB** | 111/110 | input byte |
| D2ma | **INWQ** | 111/110 | input word, end bus |
| D3ma | **INBQ** | 111/110 | input byte , end bus |
|  | | | M[4] – FPP register bank select, 0 – B, 1- A |
| M[5] – FPP register selector FA[1] |
| M[6] – toggle lower/upper half of 32-bit FPP register |
| M[7] – FPP register selector ~FA[3] |
| D4ma | **IRW** | 001/110 | IR input, mi[7:4] -> control[3:0] |
| D5ma | **IRB** | 001/110 | IR input, mi[7:4] -> control[3:0] |
| D6ma | **IRWQ** | 001/110 | IR input, mi[7:4] -> control[3:0] , end bus cycle |
| D7ma | **IRBQ** | 001/110 | IR input, mi[7:4] -> control[3:0] , end bus cycle |
| D8ma | **ISW** | 1x0/110 | PSW input word |
| D9ma | **ISB** | 1x0/110 | PSW input byte |
| DAma | **ISWQ** | 1x0/110 | PSW input, end bus cycle |
| DBma | **ISBQ** | 1x0/110 | PSW input, end bus cycle |
| DCma | **IVW** | 111/110 | IAKO |
| DDma | **IVB** | 111/110 | IAKO |
| DEma | **IVWQ** | 111/110 | IAKO, end bus cycle |
| DFma | **IVBQ** | 111/110 | IAKO, end bus cycle |

# MMU

## Регистры MMU

|  |  |  |
| --- | --- | --- |
| Address | Name | Description |
| 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 | **UPDR0** | User mode page descriptor register 0 |
| 177776028 | **UPDR1** | User mode page descriptor register 1 |
| 177776048 | **UPDR2** | User mode page descriptor register 2 |
| 177776068 | **UPDR3** | User mode page descriptor register 3 |
| 177776108 | **UPDR4** | User mode page descriptor register 4 |
| 177776128 | **UPDR5** | User mode page descriptor register 5 |
| 177776148 | **UPDR6** | User mode page descriptor register 6 |
| 177776168 | **UPDR7** | User mode page descriptor register 7 |
| 177776408 | **UPAR0** | User mode page address register 0 |
| 177776428 | **UPAR1** | User mode page address register 1 |
| 177776448 | **UPAR2** | User mode page address register 2 |
| 177776468 | **UPAR3** | User mode page address register 3 |
| 177776508 | **UPAR4** | User mode page address register 4 |
| 177776528 | **UPAR5** | User mode page address register 5 |
| 177776548 | **UPAR6** | User mode page address register 6 |
| 177776568 | **UPAR7** | User mode page address register 7 |
| 177777768 | **PSW** | Located in DC302, contains previous and current modes |

## PSW - слово состояния процессора

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name:** PSW | | | | | | | | | | | | | | | |
| **Offset:** 177777768 | | | | | | | | **Reset Value:** N/A | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| CM | | PM | | 0 | 0 | 0 | SI | PRI | | | T | N | Z | V | C |

Описание флагов слова состояния процессора **PSW** приведено в таблице:

|  |  |  |
| --- | --- | --- |
| Бит | Имя | Описание |
| 0 | **C** | Флаг переноса, устанавливается, если был перенос из старшего разряда (15-го при словных операциях и 7-го при байтовых). Операция вычитания реализовывается как сложение с инвертированным вторым операндом плюс один, поэтому при вычитании флаг переноса устанавливается если не было займа в старший разряд (не аппаратной инверсии флага переноса при вычитании). При выполнении сдвигов действуют правила описанные в соответствующих операциях |
| 1 | **V** | Флаг арифметического переполнения, при сложении вычисляется как исключающее ИЛИ переносов из старшего и предстаршего разрядов. Обычно сигнализирует о потере знака результата, используется в знаковой арифметике |
| 2 | **Z** | Флаг равенства результата нулю, устанавливается, если все биты результата ([15:0] при словных операциях и [7:0] при байтовых) |
| 3 | **N** | Флаг знака результата, равен старшему разряду (15-му для словных операций и 7-му для байтовых) |
| 4 | **T** | Флаг ловушки пошаговой отладки. |
| 7:5 | **PRI** | Биты приоритета разрешенных внешних прерываний. 7 – все запрещены 6 – прерывания с приоритетом 7 5 – прерывания с приоритетом 7, 6 4 – прерывания с приоритетом 7, 6, 5 3 – прерывания с приоритетом 7, 6, 5, 4 2 – прерывания с приоритетом 7, 6, 5, 4 1 – прерывания с приоритетом 7, 6, 5, 4 0 – прерывания с приоритетом 7, 6, 5, 4 |
| 8 | **SI** | Приостановленная инструкция, может быть прочитан-записан, зарезервирован для использования совместно с инструкциями CIS |
| 13:12 | **PM** | Предыдущий режим защиты памяти, может быть прочитан-записан, при прерывании или исключении в данные биты переписывается значение поля CM |
| 15:14 | **CM** | Текущий режим защиты памяти, может быть прочитан-записан |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mode | Name | PAR/PDR | SP | Описание |
| 00 | Kernel | Kernel | KSP | Режим ядра, привилегированный |
| 01 | Supervisor | Kernel | SSP | Режим супервизора, зарезервировано |
| 10 | - | Kernel | - | Зарезервировано, не используется |
| 11 | User | User | USP | Режим пользователя, непривилегированный |

## Регистр дескриптора страницы

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name:** PDR[7:0] – UPDR[7:0]/KPDR[7:0] | | | | | | | | | | | | | | | |
| **Offset:** 177776008/177723008 | | | | | | | | **Reset Value:** N/A | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
|  | Read & Write | | | | | | |  | Rс |  |  | RW | R & W | |  |
| 0 | PLF | | | | | | | 0 | W | 0 | 0 | ED | ACF | | 0 |

Описание битов регистра дескриптора страницы приведено в таблице:

|  |  |  |
| --- | --- | --- |
| Бит | Имя | Описание |
| 2:1 | **ACF** | **Access Control Field**, режим доступа к странице  00 - non-resident – отсутствует в памяти, любое обращение вызывает исключение 01 – read only – обращение на запись вызывает исключение 10 – unused, не используется, любое обращение вызывает исключение 11 – resident read-write, допускается чтение и запись |
| 3 | **ED** | **Expansion Direction**, если этот бит сброшен то валидной считается память 8КB страницы с адресами ниже предела указанного в поле PLF, иначе валидны адреса превышающие предел и до полного размера страницы в 8КB |
| 6 | **W** | **Written** – бит устанавливается только аппаратно, если в страницу была выполнена запись, бит сбрасывается при любой записи в регистр PDR или PDR для данной страницы. |
| 14:7 | **PLF** | **Page Length Field**, определяет размер страницы в 6410-байтных блоках |

# FPP

## Регистры FPP

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| FA[4:1] | | M[4] = 1, BANK A | | M[4] = 0, BANK B | | M[7:5] |
| FA[0] = 1 | FA[0] = 0 | FA[0]=1 | FA[0] = 0 |
| FA[4]=0 FA[3:1] | AC0 | double word 1 | double word 0 | float word 1 | float word 0 | 11x |
| AC1 | double word 1 | double word 0 | float word 1 | float word 0 | 11x |
| AC2 | double word 1 | double word 0 | float word 1 | float word 0 | 11x |
| AC3 | double word 1 | double word 0 | float word 1 | float word 0 | 11x |
| AC4 | double word 1 | double word 0 | float word 1 | float word 0 | 11x |
| AC5 | double word 1 | double word 0 | float word 1 | float word 0 | 11x |
| AC6 | double word 1 | double word 0 | float word 1 | float word 0 | 010 |
| AC7 | double word 1 | double word 0 | Instruction address | Instruction opcode | 011 |
| FA[4]=1 | 1001 |  | | FPS | | 101 |
| 1101 |  | |  | | 001 |

## FPP инструкции

| Код | Мнемоника | FNZVC | Описание |
| --- | --- | --- | --- |
| 1700008 | CFCC | ---- | Копировать флаги операций плавающей точки |
| 1700018 | SETF | ---- | Установить режим single precision, FD = 0 |
| 1700028 | SETI | ---- | Установить режим короткого целого, FL = 0 |
| 1700118 | SETD | ---- | Установить режим double precision, FD = 1 |
| 1700128 | SETL | ---- | Установить режим длинного целого, FL = 1 |
| 1701ss8 | LDFPS src | ---- | Загрузить FPS (FPP Status Register) |
| 1702dd8 | STFPS dst | --- | Сохранить FPS (FPP Status Register) |
| 1703dd8 | STST dst | --- | Сохранить FPP Exception Status (FEC and FEA) |
| 1704fd8 | CLRx fdst | 0100 | Присвоить нулевое значение |
| 1705fd8 | TSTx fdst | \*\*00 | Тестировать float/double |
| 1706fd8 | ABSx fdst | 0\*00 | Абсолютное значение float/double |
| 1707fd8 | NEGx fdst | \*\*00 | Изменить знак float/double |
| 1710acfs8 | MULx fsrc, ac | \*\*\*0 | Умножить float/double |
| 1714acfs8 | MODx fsrc, ac | \*\*\*0 | Умножить и разделить на целую и дробную части |
| 1720acfd8 | STx ac, fdst | ---- | Сохранить float/double |
| 1724acfs8 | LDx fsrc, ac | \*\*00 | Загрузить float/double |
| 1730acfs8 | SUBx fsrc, ac | \*\*\*0 | Вычитание float/double |
| 1734acfs8 | CMPx fsrc, ac | \*\*00 | Сравнить float/double |
| 1740acfs8 | ADDx fsrc, ac | \*\*\*0 | Сложить AC с fsrc |
| 1744acfs8 | DIVx fsrc, ac | \*\*\*0 | Разделить float/double |
| 1750acdd8 | STEXP ac, dst | \*\*00 | Сохранить экспоненту |
| 1754acdd8 | STCxx ac, dst | \*\*0\* | Сохранить и конвертировать float/double в int/long |
| 1760acfd8 | STCxx ac, fdst | \*\*\*0 | Сохранить и конвертировать float/double |
| 1764acfs8 | LDEXP src, ac | \*\*\*0 | Загрузить экспоненту |
| 1770acfs8 | LDCxx src, ac | \*\*\*0 | Загрузить и конвертировать int/long->float/double |
| 1774acfs8 | LDCxx fsrc, ac | \*\*\*0 | Загрузить и конвертировать float->double или double->float |

# 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 (пока обнаружено использование 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.