aba  8-bit add RegA=RegA+RegB
dbx  unsigned add RegX=RegX+RegB
aby  unsigned add RegY=RegY+RegB
adca 8-bit add with carry to RegA
adcb 8-bit add with carry to RegB
adaa 8-bit add to RegA
addb 8-bit add to RegB
addd 16-bit add to RegD
anda 8-bit logical and to RegA
andb 8-bit logical and to RegB
andcc 8-bit logical and to RegCC
asl/lsl  8-bit left shift Memory
asla/lsla 8-bit left shift RegA
asl/lslb 8-bit arith left shift RegB
asld/lslrd 16-bit left shift RegD
dsr  8-bit arith right shift Memory
dsra 8-bit arith right shift to RegA
dsrb 8-bit arith right shift to RegB
bsc  branch if carry clear
bclrb  branch clear in memory  bclrb PTT,#$01
bcs  branch if carry set
beq  branch if result is zero (Z=1)
bge  branch if signed ≥
bgnd  enter background debug mode
bgt  branch if signed >
bhi  branch if unsigned >
bhs  branch if unsigned ≥
bita 8-bit and with RegA, sets CCR
bitb 8-bit and with RegB, sets CCR
ble  branch if signed ≤
blo  branch if unsigned <
bls  branch if unsigned ≤
bit  branch if signed <
 bmi  branch if result is negative (N=1)
 bne  branch if result is nonzero (Z=0)
 bpl  branch if result is positive (N=0)
 bra  branch always
brclr  branch if bits are clear  brclr PTT,#$01,loop
brn  branch never
brset  branch if bits are set  brset PTT,#$01,loop
bset  bit set in memory  bset PTT,#$04
bbr  branch to subroutine
bvc  branch if overflow clear
bvs  branch if overflow set
call  subroutine in expanded memory
cba  8-bit compare RegA with RegB,  RegA-RegB
clc  clear carry bit, C=0
cli  clear I=0, enable interrupts
clr  8-bit memory clear
 clra  RegA clear
 clrb  RegB clear
 clv  clear overflow bit, V=0
cmpeq 8-bit compare RegA with memory
cmpeqb 8-bit compare RegB with memory
com  8-bit logical complement to memory
comateq 8-bit logical complement to RegA
comb  8-bit logical complement to RegB
cdp  16-bit compare RegD with memory
cpx  16-bit compare RegX with memory
cpy  16-bit compare RegY with memory
daa  8-bit decimal adjust accumulator
dbeq  decrement and branch if result=0  dbeq Y,loop
dbne  decrement and branch if result≠0  dbne A,loop
dec  8-bit decrement memory
decb  8-bit decrement RegB
decd  8-bit decrement RegA
des  16-bit decrement RegSP
dex  16-bit decrement RegX
dey  16-bit decrement RegY
div  RegY=(Y:D)/RegX, unsigned divide
divs  RegY=(Y:D)/RegX, signed divide
eamcs  16 by 16 signed multiply, 32-bit add
eamx 16-bit unsigned maximum in RegD
eamx 16-bit unsigned maximum in memory
emind 16-bit unsigned minimum in RegD
emimm 16-bit unsigned minimum in memory
emul  RegY:=RegY*RegD unsigned multiply
emuls  RegY:=RegY*RegD signed multiply
eora  8-bit logical exclusive or to RegA
eorb  8-bit logical exclusive or to RegB
etbl 16-bit look up and interpolation
exg  exchange register contents  exg X,Y
fdiv  unsigned sqrt div, X=(65536*D)/X
ibeq  increment and branch if result=0  ibeq Y,loop
ibne  increment and branch if result≠0  ibne A,loop
idiv 16-bit unsigned div, X=D/Y, D=remainder
idivs 16-bit signed div, X=D/Y, D=remainder
inc  8-bit increment memory
inc a 8-bit increment RegA
incb  8-bit increment RegB
ins  16-bit increment RegSP
inx  16-bit increment RegX
iny  16-bit increment RegY
jmp  jump always
jsr  jump to subroutine
leay 16-bit load memory into RegY
leax 16-bit load memory into RegX
ldy  16-bit load memory into RegY
ldx  16-bit load memory into RegX
ldd  16-bit load memory into RegD
lds  16-bit load memory into RegSP
ldx  16-bit load memory into RegX
ldy  16-bit load memory into RegY
ldab 16-bit load memory into RegA
ldba 16-bit load memory into RegB
ldsp 16-bit load memory into RegSP
ldx  16-bit load memory into RegX
leas 16-bit load effective addr to SP  leas 2,sp
leax 16-bit load effective addr to X  leax 2,x
leay 16-bit load effective addr to Y  leay 2,y
lsr  8-bit logical right shift memory
lsra  8-bit logical right shift RegA
lsrb  8-bit logical right shift RegB
lsrd  16-bit logical right shift RegD
maxa 8-bit unsigned maximum in RegA
maxm 8-bit unsigned maximum in memory
mem  determine the Fuzzy logic membership grade
mina 8-bit unsigned minimum in RegA
minm 8-bit unsigned minimum in memory
movb 8-bit move memory to memory  movb #100,PTT
The page contains a mix of text and tables discussing various machine language instructions and their meanings. The table is titled "Freescale 6812 addressing modes r is X, Y, SF, or FC" and provides examples, modes, and effective addresses for different addressing modes. The text also includes various machine instructions, such as "movw 16-bit move memory to memory movw #13, SCI BD" and "xgdx exchange RegD with RegX xgdy exchange RegD with RegY".

The page also includes a section on Freescale 6812 addressing modes, which are categorized into immediate, direct, extended, indexed, and indirect addressing modes. The table provides effective addresses for different address modes.

The page also includes a section on pseudo-ops and their meanings, such as "movw 16-bit move memory to memory movw #13, SCI BD" and "xgdx exchange RegD with RegX xgdy exchange RegD with RegY".

The page includes a table of interrupt vectors and interrupt numbers, which includes various interrupt sources such as SCI0 TDRE, RDRF, Timer overflow, Timer Channel 7, and SWI, Trap, Reset, etc. The table also includes the source of the interrupt and the arm number.

The page also includes a section on Freescale 6812 addressing modes, which are categorized into immediate, direct, extended, indexed, and indirect addressing modes. The table provides effective addresses for different address modes.

The page also includes a section on pseudo-ops and their meanings, such as "movw 16-bit move memory to memory movw #13, SCI BD" and "xgdx exchange RegD with RegX xgdy exchange RegD with RegY".

The page includes a table of interrupt vectors and interrupt numbers, which includes various interrupt sources such as SCI0 TDRE, RDRF, Timer overflow, Timer Channel 7, and SWI, Trap, Reset, etc. The table also includes the source of the interrupt and the arm number.

The page also includes a section on Freescale 6812 addressing modes, which are categorized into immediate, direct, extended, indexed, and indirect addressing modes. The table provides effective addresses for different address modes.

The page also includes a section on pseudo-ops and their meanings, such as "movw 16-bit move memory to memory movw #13, SCI BD" and "xgdx exchange RegD with RegX xgdy exchange RegD with RegY".

The page includes a table of interrupt vectors and interrupt numbers, which includes various interrupt sources such as SCI0 TDRE, RDRF, Timer overflow, Timer Channel 7, and SWI, Trap, Reset, etc. The table also includes the source of the interrupt and the arm number.

The page also includes a section on Freescale 6812 addressing modes, which are categorized into immediate, direct, extended, indexed, and indirect addressing modes. The table provides effective addresses for different address modes.

The page also includes a section on pseudo-ops and their meanings, such as "movw 16-bit move memory to memory movw #13, SCI BD" and "xgdx exchange RegD with RegX xgdy exchange RegD with RegY".

The page includes a table of interrupt vectors and interrupt numbers, which includes various interrupt sources such as SCI0 TDRE, RDRF, Timer overflow, Timer Channel 7, and SWI, Trap, Reset, etc. The table also includes the source of the interrupt and the arm number.

The page also includes a section on Freescale 6812 addressing modes, which are categorized into immediate, direct, extended, indexed, and indirect addressing modes. The table provides effective addresses for different address modes.

The page also includes a section on pseudo-ops and their meanings, such as "movw 16-bit move memory to memory movw #13, SCI BD" and "xgdx exchange RegD with RegX xgdy exchange RegD with RegY".

The page includes a table of interrupt vectors and interrupt numbers, which includes various interrupt sources such as SCI0 TDRE, RDRF, Timer overflow, Timer Channel 7, and SWI, Trap, Reset, etc. The table also includes the source of the interrupt and the arm number.

The page also includes a section on Freescale 6812 addressing modes, which are categorized into immediate, direct, extended, indexed, and indirect addressing modes. The table provides effective addresses for different address modes.

The page also includes a section on pseudo-ops and their meanings, such as "movw 16-bit move memory to memory movw #13, SCI BD" and "xgdx exchange RegD with RegX xgdy exchange RegD with RegY".

The page includes a table of interrupt vectors and interrupt numbers, which includes various interrupt sources such as SCI0 TDRE, RDRF, Timer overflow, Timer Channel 7, and SWI, Trap, Reset, etc. The table also includes the source of the interrupt and the arm number.

The page also includes a section on Freescale 6812 addressing modes, which are categorized into immediate, direct, extended, indexed, and indirect addressing modes. The table provides effective addresses for different address modes.

The page also includes a section on pseudo-ops and their meanings, such as "movw 16-bit move memory to memory movw #13, SCI BD" and "xgdx exchange RegD with RegX xgdy exchange RegD with RegY".

The page includes a table of interrupt vectors and interrupt numbers, which includes various interrupt sources such as SCI0 TDRE, RDRF, Timer overflow, Timer Channel 7, and SWI, Trap, Reset, etc. The table also includes the source of the interrupt and the arm number.

The page also includes a section on Freescale 6812 addressing modes, which are categorized into immediate, direct, extended, indexed, and indirect addressing modes. The table provides effective addresses for different address modes.

The page also includes a section on pseudo-ops and their meanings, such as "movw 16-bit move memory to memory movw #13, SCI BD" and "xgdx exchange RegD with RegX xgdy exchange RegD with RegY".

The page includes a table of interrupt vectors and interrupt numbers, which includes various interrupt sources such as SCI0 TDRE, RDRF, Timer overflow, Timer Channel 7, and SWI, Trap, Reset, etc. The table also includes the source of the interrupt and the arm number.

The page also includes a section on Freescale 6812 addressing modes, which are categorized into immediate, direct, extended, indexed, and indirect addressing modes. The table provides effective addresses for different address modes.

The page also includes a section on pseudo-ops and their meanings, such as "movw 16-bit move memory to memory movw #13, SCI BD" and "xgdx exchange RegD with RegX xgdy exchange RegD with RegY".

The page includes a table of interrupt vectors and interrupt numbers, which includes various interrupt sources such as SCI0 TDRE, RDRF, Timer overflow, Timer Channel 7, and SWI, Trap, Reset, etc. The table also includes the source of the interrupt and the arm number.

The page also includes a section on Freescale 6812 addressing modes, which are categorized into immediate, direct, extended, indexed, and indirect addressing modes. The table provides effective addresses for different address modes.

The page also includes a section on pseudo-ops and their meanings, such as "movw 16-bit move memory to memory movw #13, SCI BD" and "xgdx exchange RegD with RegX xgdy exchange RegD with RegY".

The page includes a table of interrupt vectors and interrupt numbers, which includes various interrupt sources such as SCI0 TDRE, RDRF, Timer overflow, Timer Channel 7, and SWI, Trap, Reset, etc. The table also includes the source of the interrupt and the arm number.

The page also includes a section on Freescale 6812 addressing modes, which are categorized into immediate, direct, extended, indexed, and indirect addressing modes. The table provides effective addresses for different address modes.
### Address

<table>
<thead>
<tr>
<th>Address</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0040</td>
<td>IOS7</td>
<td>IOS6</td>
<td>IOS5</td>
<td>IOS4</td>
<td>IOS3</td>
<td>IOS2</td>
<td>IOS1</td>
<td>IOS0</td>
<td>TIOS</td>
</tr>
<tr>
<td>$0044-5</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$0046</td>
<td>TEN</td>
<td>TSWAI</td>
<td>TSFRZ</td>
<td>TFFCA</td>
<td>C71</td>
<td>C51</td>
<td>C41</td>
<td>C21</td>
<td>C11</td>
</tr>
<tr>
<td>$004D</td>
<td>TOI</td>
<td>0</td>
<td>PUPT</td>
<td>RDPT</td>
<td>TCRT</td>
<td>PR2</td>
<td>PR1</td>
<td>PR0</td>
<td>TSCR1</td>
</tr>
<tr>
<td>$004E</td>
<td>C7F</td>
<td>C6F</td>
<td>C5F</td>
<td>C4F</td>
<td>C3F</td>
<td>C2F</td>
<td>C1F</td>
<td>C0F</td>
<td>TFLG1</td>
</tr>
<tr>
<td>$004F</td>
<td>TOF</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>TFLG2</td>
<td></td>
</tr>
<tr>
<td>$0050-1</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td>TC0</td>
</tr>
<tr>
<td>$0052-3</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td>TC1</td>
</tr>
<tr>
<td>$0054-5</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td>TC2</td>
</tr>
<tr>
<td>$0056-7</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td>TC3</td>
</tr>
<tr>
<td>$0058-9</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td>TC4</td>
</tr>
<tr>
<td>$005A-B</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td>TC5</td>
</tr>
<tr>
<td>$005C-D</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td>TC6</td>
</tr>
<tr>
<td>$005E-F</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td>TC7</td>
</tr>
<tr>
<td>$0082</td>
<td>ADPU</td>
<td>AFFC</td>
<td>ASWAI</td>
<td>ETRIGLE</td>
<td>ETRIGP</td>
<td>ETRIG</td>
<td>ASCIE</td>
<td>ASCIF</td>
<td>ATD0CTL2</td>
</tr>
<tr>
<td>$0083</td>
<td>S8C</td>
<td>S4C</td>
<td>S2C</td>
<td>S1C</td>
<td>FIFO</td>
<td>FRZ1</td>
<td>FRZ0</td>
<td></td>
<td>ATD0CTL3</td>
</tr>
<tr>
<td>$0084</td>
<td>SRE8S</td>
<td>SMP1</td>
<td>SMP0</td>
<td>PSR4</td>
<td>PRS3</td>
<td>PRS2</td>
<td>PRS1</td>
<td>PRS0</td>
<td>ATD0CTL4</td>
</tr>
<tr>
<td>$0085</td>
<td>DJM</td>
<td>DSGN</td>
<td>SCAN</td>
<td>MULT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ATD0CTL5</td>
</tr>
<tr>
<td>$0086</td>
<td>SCF</td>
<td>0</td>
<td>ETORF</td>
<td>FIFOR</td>
<td>0</td>
<td>CC</td>
<td>CB</td>
<td>CA</td>
<td>ATD0STAT0</td>
</tr>
<tr>
<td>$008B</td>
<td>CCF7</td>
<td>CCF6</td>
<td>CCF5</td>
<td>CCF4</td>
<td>CCFF2</td>
<td>CCFF1</td>
<td>CCFF0</td>
<td></td>
<td>ATD0STAT1</td>
</tr>
<tr>
<td>$008D</td>
<td>Bit 7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
<td>ATD0DIEN</td>
</tr>
<tr>
<td>$008F</td>
<td>PAD07</td>
<td>PAD06</td>
<td>PAD05</td>
<td>PAD04</td>
<td>PAD03</td>
<td>PAD02</td>
<td>PAD01</td>
<td>PAD00</td>
<td>PORTAD0</td>
</tr>
<tr>
<td>$0090-1</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td>ATD0DR0</td>
</tr>
<tr>
<td>$0092-3</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td>ATD0DR1</td>
</tr>
<tr>
<td>$0094-5</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td>ATD0DR2</td>
</tr>
<tr>
<td>$0096-7</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td>ATD0DR3</td>
</tr>
<tr>
<td>$0098-9</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td>ATD0DR4</td>
</tr>
<tr>
<td>$009A-B</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td>ATD0DR5</td>
</tr>
<tr>
<td>$009C-D</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td>ATD0DR6</td>
</tr>
<tr>
<td>$009E-F</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td>ATD0DR7</td>
</tr>
<tr>
<td>$00C9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>SBR12</td>
<td>SBR11</td>
<td>SBR10</td>
<td>SBR0</td>
<td>SC10BD</td>
<td></td>
</tr>
<tr>
<td>$00CA</td>
<td>LOOPS</td>
<td>SCI1SWAI</td>
<td>RSRC</td>
<td>M</td>
<td>WAKE</td>
<td>ILT</td>
<td>PE</td>
<td>PT</td>
<td>SCI1CR1</td>
</tr>
<tr>
<td>$00CB</td>
<td>TIE</td>
<td>TCIE</td>
<td>RIE</td>
<td>ILIE</td>
<td>TE</td>
<td>RE</td>
<td>REWU</td>
<td>SBK</td>
<td>SCI1CR2</td>
</tr>
<tr>
<td>$00CC</td>
<td>TDRE</td>
<td>TC</td>
<td>RDRF</td>
<td>IDLE</td>
<td>OR</td>
<td>NF</td>
<td>FE</td>
<td>PF</td>
<td>SCI1SR1</td>
</tr>
<tr>
<td>$00CD</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>BRK13</td>
<td>TXDIR</td>
<td>RAF</td>
<td>SCI1SR2</td>
<td></td>
</tr>
<tr>
<td>$00CF</td>
<td>R7/T7</td>
<td>R6/T6</td>
<td>R5/T5</td>
<td>R4/T4</td>
<td>R3/T3</td>
<td>R2/T2</td>
<td>R1/T1</td>
<td>R0/T0</td>
<td>SCI0DRL</td>
</tr>
<tr>
<td>$00D0-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>SBR12</td>
<td>SBR11</td>
<td>SBR10</td>
<td>SBR0</td>
<td>SCI1BD</td>
<td></td>
</tr>
<tr>
<td>$00D2</td>
<td>LOOPS</td>
<td>SCI1SWAI</td>
<td>RSRC</td>
<td>M</td>
<td>WAKE</td>
<td>ILT</td>
<td>PE</td>
<td>PT</td>
<td>SCI1CR1</td>
</tr>
<tr>
<td>$00D3</td>
<td>TIE</td>
<td>TCIE</td>
<td>RIE</td>
<td>ILIE</td>
<td>TE</td>
<td>RE</td>
<td>REWU</td>
<td>SBK</td>
<td>SCI1CR2</td>
</tr>
<tr>
<td>$00D4</td>
<td>TDRE</td>
<td>TC</td>
<td>RDRF</td>
<td>IDLE</td>
<td>OR</td>
<td>NF</td>
<td>FE</td>
<td>PF</td>
<td>SCI1SR1</td>
</tr>
<tr>
<td>$00D5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>BRK13</td>
<td>TXDIR</td>
<td>RAF</td>
<td>SCI1SR2</td>
<td></td>
</tr>
<tr>
<td>$00D7</td>
<td>R7/T7</td>
<td>R6/T6</td>
<td>R5/T5</td>
<td>R4/T4</td>
<td>R3/T3</td>
<td>R2/T2</td>
<td>R1/T1</td>
<td>R0/T0</td>
<td>SCI1DRL</td>
</tr>
<tr>
<td>$00D9</td>
<td>PJ7</td>
<td>PT6</td>
<td>PT5</td>
<td>PT4</td>
<td>PT3</td>
<td>PT2</td>
<td>PT1</td>
<td>PT0</td>
<td></td>
</tr>
<tr>
<td>$0242</td>
<td>DDR7</td>
<td>DDR6</td>
<td>DDR5</td>
<td>DDR4</td>
<td>DDR3</td>
<td>DDR2</td>
<td>DDR1</td>
<td>DDR0</td>
<td>DDRT</td>
</tr>
<tr>
<td>$0248</td>
<td>PS7</td>
<td>PS6</td>
<td>PS5</td>
<td>PS4</td>
<td>PS3</td>
<td>PS2</td>
<td>PS1</td>
<td>PS0</td>
<td>PTS</td>
</tr>
<tr>
<td>$0252</td>
<td>DRRM7</td>
<td>DRRM6</td>
<td>DRRM5</td>
<td>DRRM4</td>
<td>DRRM3</td>
<td>DRRM2</td>
<td>DRRM1</td>
<td>DRRM0</td>
<td>DRRM</td>
</tr>
<tr>
<td>$0258</td>
<td>PP7</td>
<td>PP6</td>
<td>PP5</td>
<td>PP4</td>
<td>PP3</td>
<td>PP2</td>
<td>PP1</td>
<td>PP0</td>
<td>PTP</td>
</tr>
<tr>
<td>$0262</td>
<td>DDRH7</td>
<td>DDRH6</td>
<td>DDRH5</td>
<td>DDRH4</td>
<td>DDRH3</td>
<td>DDRH2</td>
<td>DDRH1</td>
<td>DDRH0</td>
<td>DDRH</td>
</tr>
<tr>
<td>$0268</td>
<td>PJ7</td>
<td>PJ6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>PJ1</td>
<td>PJ0</td>
<td>PT</td>
<td></td>
</tr>
<tr>
<td>$026A</td>
<td>DDRJ7</td>
<td>DDRJ6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>DDRJ1</td>
<td>DDRJ0</td>
<td>DDRJ</td>
<td></td>
</tr>
</tbody>
</table>

**TSCR1** is the first 8-bit timer control register
- bit 7 **TEN**: 1 allows the timer to function normally, 0 means disable timer including **TCNT**
- **TIOS** is the 8-bit output compare select register, one bit for each channel (1 = output compare, 0 = input capture)
- **TIE** is the 8-bit output compare arm register, one bit for each channel (1 = armed, 0 = disarmed)
**TSCR2** is the second 8-bit timer control register

bits 2,1,0 are **PR2, PR1, PR0**, which select the rate, let $n$ be the 3-bit number formed by **PR2, PR1, PR0**

without PLL TCNT is $8MHz/2^n$, with PLL TCNT is $24MHz/2^n$, $n$ ranges from 0 to 7

<table>
<thead>
<tr>
<th>PR2</th>
<th>PR1</th>
<th>PR0</th>
<th>Divide by</th>
<th>E = 8 MHz</th>
<th>E = 24 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TCNT</td>
<td>TCNT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>period</td>
<td>frequency</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TCNT</td>
<td>TCNT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>period</td>
<td>frequency</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>125 ns</td>
<td>8 MHz</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>250 ns</td>
<td>4 MHz</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>500 ns</td>
<td>2 MHz</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>1 µs</td>
<td>1 MHz</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>2 µs</td>
<td>500 kHz</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>32</td>
<td>4 µs</td>
<td>250 kHz</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>64</td>
<td>8 µs</td>
<td>125 kHz</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>128</td>
<td>16 µs</td>
<td>62.5 kHz</td>
</tr>
</tbody>
</table>

**SCI0DRL** 8-bit SCI data register

**SCI0BD** is 16-bit SCI0 baud rate register, let $n$ be the 13-bit number  Baud rate is $E\text{_clk}/n/16$

**SCI0CR1** is 8-bit SCI0 control register

bit 4 M, Mode, 0 = One start, eight data, one stop bit, 1 = One start, eight data, ninth data, one stop bit

**SCI0CR2** is 8-bit SCI0 control register

bit 7 TIE, Transmit Interrupt Enable, 0 = TDRE interrupts disabled, 1 = interrupt whenever TDRE set
bit 5 RIE, Receiver Interrupt Enable, 0 = RDRF interrupts disabled, 1 = interrupt whenever RDRF set
bit 3 TE, Transmitter Enable, 0 = Transmitter disabled, 1 = SCI transmit logic is enabled
bit 2 RE, Receiver Enable, 0 = Receiver disabled, 1 = Enables the SCI receive circuitry.

**SCI0SR1** is 8-bit SCI0 status register

bit 7 TDRE, Transmit Data Register Empty Flag
Set if transmit data can be written to **SCI0DRL**
Cleared by **SCI0SR1** read with TDRE set followed by **SCI0DRL** write.

bit 5 RDRF, Receive Data Register Full
set if a received character is ready to be read from **SCI0DRL**
Clear the RDRF flag by reading **SCI0SR1** with RDRF set and then reading **SCI0DRL**.