

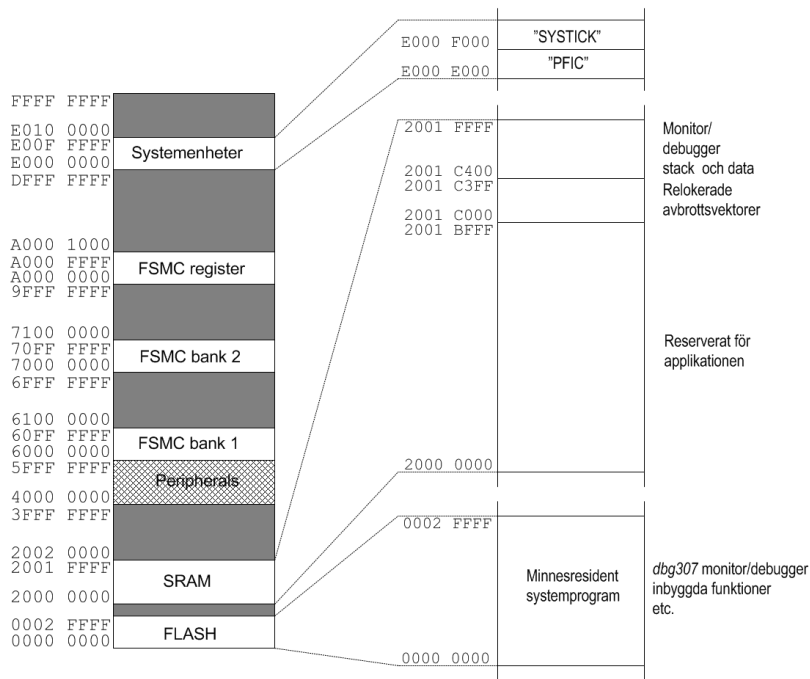
Quick Guide MD307

Detta häfte får användas under tentamen i kursen "Maskinorienterad programmering" under förutsättning att inga egna anteckningar gjorts.
Ev. rättelser och/eller kompletteringar bifogas tentamenstes.

Institutionen för Data och informationsteknik
Chalmers 2026-03-15

Minnesdisposition	2
Periferienheter	3
Registeruppsättning och kompilatorkonventioner	4
RISC-V instruktionsuppsättning	5
RV32I base	5
RV32M	7
RV32 (Ziscr)	8
RV32F	8
Typiska assemblerdirektiv	9
Undantagshantering	10
Behörighetsnivåer MD307	10
Speciella register MD307 (CSR)	10
PFIC, Programmable Fast Interrupt Controller	12
EXTI External Interrupt	13
AFIO EXTI Control	13
Vektortabell	14
Systemkretsar och periferienheter	16
Räknarkrets SYSTICK	16
GPIO	17
TIMER6/TIMER7	18
USART	19
C Quick Reference Guide	22

MINNESDISPOSITION



Följande adresser används för inbyggda funktioner i *dbg307*:

```

0x08000200: dbg_restart
0x08000204: dbg_clockinit
0x08000208: dbg_portinit
0x08000210: dbg_outchar
0x08000214: dbg_tstchar
0x08000230: dbg_tft_init
0x08000234: dbg_tft_lcd_ellipse
0x08000238: dbg_tft_lcd_rect
0x0800023C: dbg_tft_lcd_line
0x08000244: dbg_tft_lcd_pixel
0x08000248: dbg_delay_ms
0x0800024C: dbg_tft_tp_scan
0x08000250: dbg_tft_tp_getpos

```

Beskrivning av minnesresidenta funktioner i *dbg307* (*dbg307.h/libdbg307.a*)

void dbg_restart(void);

Funktionen återstartar *dbg307* från prompter.

void dbg_clockinit(void);

Funktionen konfigurerar systemets klockor och initierar PLL för 144MHz klockfrekvens, startar klockor för TIMER6, TIMER7, GPIO A, GPIO D, GPIO E, AFIO och USART1(SPI1).

void dbg_portinit(void);

Konfigurerar USART1 (115200 baud) mottagardel och sändardel för "polling".

Konfigurerar port D och E ("input floating")

void dbg_outchar(unsigned char c);

Matar ut ett tecken via USART1. Förutsätter att *clockinit* och *portinit* utförts.

unsigned char dbg_tstchar(void);

Kontrollerar om tecken mottagits av USART1, returnerar 1 så fall detta. Om inget tecken mottagits returneras 0.

int dbg_tft_init(int option)

initierar TFT-display på laborationskortet PTB-111. option är ett fält som bestämmer hur displayen ska konfigureras, exakt en displaytyp måste anges:

```
#define TFT_INIT_ILI9488 1 dvs. 3.2" display
```

```
#define TFT_INIT_ST7796S 2 dvs. 4.0" display
```

Man kan samtidigt konfigurera touch-funktionen hos displayen:

```
#define TFT_INIT_TOUCH 0x10
```

Funktionen returnerar 0 om konfiguration inte kunde slutföras, respektive 1 om konfigurationen lyckades. Samtidigt sätts då displayens bakgrundsfärg till gul.

void dbg_tft_lcd_ellipse(int xc, int yc, int rx, int ry, int colour, int fill);

Funktionen ritar en ellips med centrum i *xc/yc*, horisontell diameter *rx* och vertikal diameter *ry* med färgen *colour* (RGB16,5:6:5).

Ellipsen fylls med samma färg om *fill* är 1, om *fill* är 0 ritas bara ellipsen.

void dbg_tft_lcd_rect(int xc, int yc, int x1, int y1, int colour, int fill);

Funktionen ritar en rektangel med övre vänstra hörnet i *xc/yc*, längden *x1* och höjden *y1* med färgen *colour* (RGB16,5:6:5). Ellipsen fylls med samma färg om *fill* är 1, om *fill* är 0 ritas bara rektangeln.

void dbg_tft_lcd_line(int x1, int y1, int x2, int y2, int colour);

Funktionen ritar en linje med bredden 1 pixel, från punkten *x1/y1* till punkten *x2/y2*, med färgen *colour* (RGB16,5:6:5).

void dbg_tft_lcd_pixel(int x, int y, int colour);

Funktionen ritar en pixel i position *x/y* med färgen *colour* (RGB16,5:6:5).

void dbg_delay_ms(unsigned int ms);

Blockerande fördröjning minst *ms* millisekunder. Funktionen använder TIMER7.

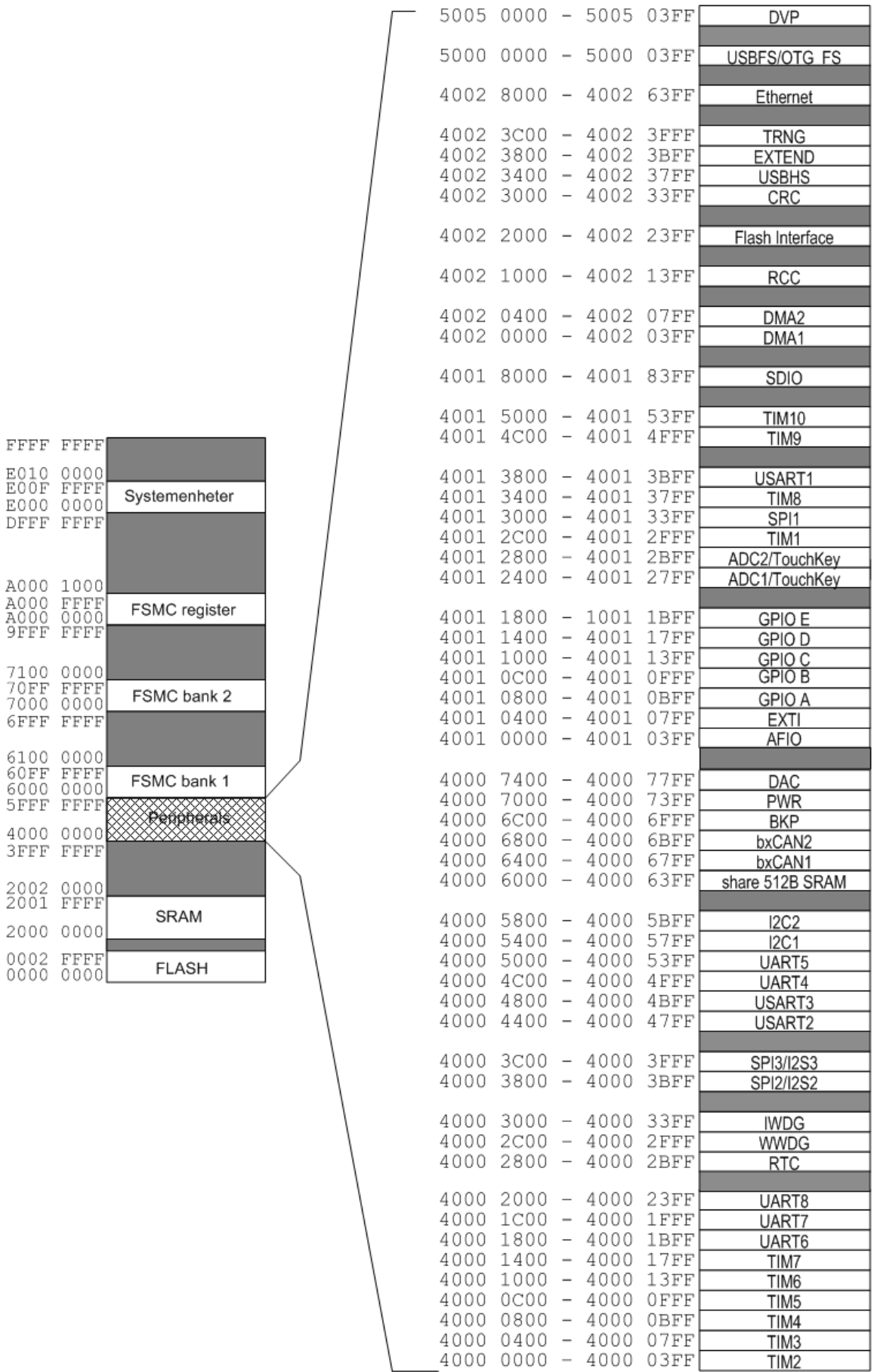
int dbg_tft_tp_scan(int tp);

Funktionen returnerar 1 om touch-panelens scan-funktion utförts korrekt, 0 annars. Parametern *tp* används för speciella kalibreringsändamål och ska normalt vara 0.

int dbg_tft_tp_getpos(int *x, int *y);

Funktionen returnerar 1 om displayens touch-logik registrerat en beröringsposition, då innehåller också *x* och *y* koordinaterna för beröringen. *dbg_tft_tp_scan* måste först ha utförts korrekt

PERIFERIEHETER



REGISTERUPPSÄTTNING OCH KOMPILATORKONVENTIONER

Register	ABI Namn	Användning	Sparas av...
x0	zero	Zero constant	—
x1	ra	Return address	Anropad
x2	sp	Stack pointer	Anropad
x3	gp	Global pointer	—
x4	tp	Thread pointer	—
x5-x7	t0-t2	Temporaries	Anropande
x8	s0 / fp	Saved / frame pointer	Anropad
x9	s1	Saved register	Anropad
x10-x11	a0-a1	Fn args/return values	Anropande
x12-x17	a2-a7	Fn args	Anropande
x18-x27	s2-s11	Saved registers	Anropad
x28-x31	t3-t6	Temporaries	Anropande
f0-7	ft0-7	FP temporaries	Anropande
f8-9	fs0-1	FP saved registers	Anropad
f10-11	fa0-1	FP args/return values	Anropande
f12-17	fa2-7	FP args	Anropande
f18-27	fs2-11	FP saved registers	Anropad
f28-31	ft8-11	FP temporaries	Anropande

Register	ABI Namn	Användning	Sparas av:
x0	zero	alltid 0, kan inte ändras	-
x1	ra	länkregister (returadress)	anropande
x2	sp	stackpekare	anropad
x3	gp	global pekare (basadress till globala variabler)	-
x4	tp	trådpekare (basadress till aktuell tråd)	-
x5-x7	t0-t2	Dessa register är avsedda som temporära register. Om dom används måste dom sparas och återställas av den anropade (<i>callee</i>) funktionen	anropande
x8	s0 (fp)	sparad pekare till aktiveringspost	anropad
x9	s1	sparat register	anropad
x10-x11	a0-a1	Parametrar resp. returvärden	anropande
x12-x17	a2-a7	Parameters	anropande
x18-x27	s2-s11	sparade register	anropad
x28-x31	t3-t6	temporaries	anropande
f0-f7	ft0-ft7	floating point temporaries	anropande
f8-f9	fs0-fs1	floating point saved registers	anropad
f10-f11	fa0-fa1	floating point Parameters/Return values	anropande
f12-f17	fa2-fa7	floating point Parameters	anropande
f18-f27	fs2-fs11	floating point saved registers	anropad
f28-f31	ft8-ft11	floating point temporaries	anropande

Aktiveringspost:

Inträde, "prolog"	Utträde, "epilog"
<ol style="list-style-type: none"> Reservera utrymme på stacken Spara register som används i funktionen (se tabellen ovan, "Sparas av=anropad"). Det innebär som minst register s0. Sätt upp pekare till den nya aktiveringsposten, dvs. s0. Om funktionen har parametrar sparas kopior av dessa register i aktiveringsposten (spill parametrar). 	<ol style="list-style-type: none"> Återställ register från stacken. Som minst register s0. Frigör det reserverade stackutrymmet

ALU operationer:

Add	add rd,rs1,rs2	rd = rs1 + rs2
Subtract	sub rd,rs1,rs2	rd = rs1 - rs2
Exclusive bitwise or	xor rd,rs1,rs2	rd = rs1 ^ rs2
Bitwise or	or rd,rs1,rs2	rd = rs1 rs2
Bitwise and	and rd,rs1,rs2	rd = rs1 & rs2
Shift left logical	sll rd,rs1,rs2	rd = rs1 << rs2
Shift right logical	srl rd,rs1,rs2	rd = rs1 >> rs2
Shift right arithmetical	sra rd,rs1,rs2	rd = rs1 >> rs2
Set if less than	slt rd,rs1,rs2	rd = (rs1 < rs2)?1:0
Set if less than (unsigned)	sltu rd,rs1,rs2	rd = (rs1 < rs2)?1:0
Add immediate	addi rd,rs1,imm	rd = rs1 + simm12
Exclusive bitwise or immediate	xori rd,rs1,imm	rd = rs1 ^ simm12
Bitwise or immediate	ori rd,rs1,imm	rd = rs1 simm12
Bitwise and immediate	andi rd,rs1,imm	rd = rs1 & simm12
Shift left logical immediate	slli rd,rs1,imm	rd = rs1 << uimm5
Shift right logical immediate	srlr rd,rs1,imm	rd = rs1 >> uimm5
Shift right Arith immediate	srair rd,rs1,imm	rd = rs1 >> uimm5
Set if less than immediate	slti rd,rs1,imm	rd = (rs1 < simm12)?1:0
Set if less than immediate (unsigned)	sltiu rd,rs1,imm	rd = (rs1 < simm12)?1:0

Pseudo instruktioner:

Pseudo form	Maskininstruktion	Beskrivning
mv rd,rs	addi rd,rs,0	Copy register
not rd,rs	xori rd,rs,-1	One's complement
neg rd,rs	sub rd,x0,rs	Two's complement
negw rd,rs	subw rd,x0,rs	Two's complement word
sext.w rd,rs	addiw rd,rs,0	Sign extend word
seqz rd,rs	sltiu rd,rs,1	Set if rs == zero
snez rd,rs	sltu rd,x0,rs	Set if rs != zero
sltz rd,rs	slt rd,rs,x0	Set if rs < zero
sgtz rd,rs	slt rd,x0,rs	Set if rs > zero

Programflödesoperationer:

Jump and link	jal ra,offset anm: assemblerformen är: jal ra,target där offset beräknats av assemblern	ra=PC+4; PC= PC+offset
Jump register and link	jalr ra,rs,imm	ra=PC+4; PC= PC+rs+imm

Pseudo instruktioner:

Pseudo form	Maskininstruktion	Beskrivning
j offset	jal x0,offset	Jump
jal offset	jal x1,offset	Jump and link
jr rs	jalr x0,rs,0	Jump register
jalr rs	jalr x1,rs,0	Jump and link register
ret	jalr x0,x1,0	Return from subroutine
call target	auipc x1,offset[31:12] jalr x1,x1,offset[11:0]	Call far-away subroutine
tail target	auipc x6,offset[31:12] jalr x0,x6,offset[11:0]	Jump far-away

Branch on equal	beq rs1,rs2,target	if(rs1 == rs2) PC = target
Branch on not equal	bne rs1,rs2,target	if(rs1 != rs2) PC = target
Branch if less, signed	blt rs1,rs2,target	if(rs1 < rs2) PC = target
Branch if greater or equal, signed	bge rs1,rs2,target	if(rs1 >= rs2) PC = target
Branch if less, unsigned	bltu rs1,rs2,target	if(rs1 < rs2) PC = target
Branch if greater or equal, unsigned	bgeu rs1,rs2,target	if(rs1 >= rs2) PC = target

C-operator	Betydelse	Datatyp	Instruktion
==	Lika	signed/unsigned	beq rs1,rs2,trueLabel
!=	Skild från	signed/unsigned	bne rs1,rs2,trueLabel
<	Mindre än	signed	blt rs1,rs2,trueLabel
		unsigned	bltu rs1,rs2,trueLabel
>=	Större än eller lika	signed	bge rs1,rs2,trueLabel
		unsigned	bgeu rs1,rs2,trueLabel

Villkor		Komplementärt villkor	
==	Lika	!=	Skild från
!=	Skild från	==	Lika
<	Mindre än	>=	Större än eller lika
>=	Större än eller lika	<	Mindre än
>	Större än	<=	Mindre än eller lika
<=	Mindre än eller lika	>	Större än

Pseudo form	Maskininstruktion	Beskrivning
beqz rs,trueLabel	beq rs,x0,trueLabel	Branch if rs= zero
bnez rs,trueLabel	bne rs,x0,trueLabel	Branch if rs!= zero
blez rs,trueLabel	bge x0,rs,trueLabel	Branch if rs ≤ zero
bgez rs,trueLabel	bge rs,x0,trueLabel	Branch if rs ≥ zero
bltz rs,trueLabel	blt rs,x0,trueLabel	Branch if rs < zero
bgtz rs,trueLabel	blt x0,rs,trueLabel	Branch if rs > zero
bgt rs,rt,trueLabel	blt rt,rs,trueLabel	Branch if rs > rt, signed
ble rs,rt,trueLabel	bge rt,rs,trueLabel	Branch if rs ≤ rt, signed
bgtu rs,rt,trueLabel	bltu rt,rs,trueLabel	Branch if rs > rt, unsigned
bleu rs,rt,trueLabel	bgeu rt,rs,trueLabel	Branch if rs ≤ rt, unsigned

Översikt – relationsoperatorer och branch-instruktioner

C-operator	Betydelse	Datatyp	Instruktion	Komplementinstruktion
==	Lika med	signed/unsigned	beq	bne
!=	Skild från	signed/unsigned	bne	beq
<	Mindre än	signed	blt	bge
		unsigned	bltu	bgeu
<=	Mindre än eller lika	signed	ble	bgt
		unsigned	bleu	bgtu
>	Större än	signed	bgt	ble
		unsigned	bgtu	bleu
>=	Större än eller lika	signed	bge	blt
		unsigned	bgeu	bltu

Diverse:

Environment Call	ecall	Transfer control to OS
Environment Break	ebreak	Transfer control to debugger
Return from exception	mret	Restore status from CSR's
Memory ordering fence	fence	

RV32M

Multiply	mul rd,rs1,rs2	rd = (rs1 * rs2)[31:0]
Multiply high word result (signed/signed)	mulh rd,rs1,rs2	rd = (rs1 * rs2)[63:32]
Multiply high word result (unsigned/unsigned)	mulu rd,rs1,rs2	rd = (rs1 * rs2)[63:32]
Multiply high word result (signed/unsigned)	mulsu rd,rs1,rs2	rd = (rs1 * rs2)[63:32]
Divide	div rd,rs1,rs2	rd = rs1/rs2
Divide unsigned	divu rd,rs1,rs2	rd = rs1/rs2
Remainder	rem rd,rs1,rs2	rd = rs1 % rs2
Remainder unsigned	remu rd,rs1,rs2	rd = rs1 % rs2

RV32 (Ziscr)

Atomic Read/Write CSR	csrrw rd,csr,rs	rd = csr; csr = rs
	csrrwi rd,csr,imm	rd = csr; csr = simm5
Atomic Read and Set Bits in CSR	csrrs rd,csr,rs	rd = csr; csr = setbits(rs)
	csrrsi rd,csr,imm	rd = csr; csr = setbits(simm5)
Atomic Read and Clear Bits in CSR	csrrc rd,csr,rs	rd = csr; csr = clearbits(rs)
	csrrci rd,csr,imm	rd = csr; csr = clearbits(simm5)

Pseudo form	Maskininstruktion	Beskrivning
csrw csr,rs	csrrw zero,csr,rs	csr = rs
csrr rd,csr	csrrs rd,csr,zero	rd = csr
csrs csr,rs	csrrs zero,csr,rs	csr = setbits(rs)
csrc csr,rs	csrrc zero,csr,rs	csr = clearbits(rs)

RV32F

Flt Load Word	flw fd,imm(rs)	fd = M[rs + simm12]
Flt Store Word	fsw fs,imm(rs)	fs = M[rs + simm12]
Flt Fused Mul-Add	fmadd.s fd,fs1,fs2,fs3	fd = fs1 * fs2 + fs3
Flt Fused Mul-Sub	fmsub.s fd,fs1,fs2,fs3	fd = fs1 * fs2 - fs3
Flt Neg Fused Mul-Add	fnmadd.s fd,fs1,fs2,fs3	fd = -fs1 * fs2 + fs3
Flt Neg Fused Mul-Sub	fnmsub.s fd,fs1,fs2,fs3	fd = -fs1 * fs2 - fs3
Flt Add	fadd.s fd,fs1,fs2	fd = fs1 + fs2
Flt Sub	fsub.s fd,fs1,fs2	fd = fs1 - fs2
Flt Mul	fmul.s fd,fs1,fs2	fd = fs1 * fs2
Flt Div	fdiv.s fd,fs1,fs2	fd = fs1 / fs2
Flt Square Root	fsqrt.s fd,fs1	fd = sqrt(fs1)
Flt Sign Injection	fsgnj.s fd,fs1,fs2	fd = abs(fs1) * sgn(fs2)
Flt Sign Neg Injection	fsgnjn.s fd,fs1,fs2	fd = abs(fs1) * -sgn(fs2)
Flt Sign Xor Injection	fsgnjx.s fd,fs1,fs2	fd = fs1 * sgn(fs2)
Flt Minimum	fmin.s fd,fs1,fs2	fd = min(fs1, fs2)
Flt Maximum	fmax.s fd,fs1,fs2	fd = max(fs1, fs2)
Flt Conv from Sign Int	fcvt.s.w rd,fs	rd = (float) fs
Flt Conv from Uns Int	fcvt.s.wu rd,fs	rd = (float) fs
Flt Convert to Int	fcvt.w.s rd,fs	rd = (int32_t) fs
Flt Convert to Int	fcvt.wu.s rd,fs	rd = (uint32_t) fs
Move Float to Int	fmv.x.w rd,fs	rd = *((int*) &fs)
Move Int to Float	fmv.w.x rd,fs	rd = *((float*) &fs)
Float Equality	feq.s fd,fs1,fs2	fd = (fs1 == fs2) ? 1 : 0
Float Less Than	flt.s fd,fs1,fs2	fd = (fs1 < fs2) ? 1 : 0
Float Less / Equal	fle.s fd,fs1,fs2	fd = (fs1 <= fs2) ? 1 : 0
Float Classify	fclass.s rd,fs1	fd = 0..9

Pseudo instruktioner:

Pseudo form	Maskininstruktion	Beskrivning
flw rd,symbol,rt	auipc rt,symbol[31:12] flw rd,symbol[11:0](rt)	Load global floating point single Note: common name
fld rd,symbol,rt	auipc rt,symbol[31:12] fld rd,symbol[11:0](rt)	Load global floating point double Note: common name
fsw rd,symbol,rt	auipc rt,symbol[31:12] fsw rd,symbol[11:0](rt)	Store global floating point single Note: common name
fsd rd,symbol,rt	auipc rt,symbol[31:12] fsd rd,symbol[11:0](rt)	Store global floating point double Note: common name
fmv.s rd,rs	fsgnj.s rd,rs,rs	Copy single-precision register
fabs.s rd,rs	fsgnjx.s rd,rs,rs	Single-precision absolute value
fneg.s rd,rs	fsgnjn.s rd,rs,rs	Single-precision negate
fmv.d rd,rs	fsgnj.d rd,rs,rs	Copy double-precision register
fabs.d rd,rs	fsgnjx.d rd,rs,rs	Double-precision absolute value
fneg.d rd,rs	fsgnjn.d rd,rs,rs	Double-precision negate

TYPISKA ASSEMBLERDIREKTIV

Listan av assemblerdirektiv som kan användas är omfattande och beror samtidigt på den använda utvecklingsmiljön. Här sammanfattar vi de vanligaste.

Direktiv	Förklaring
L: innebär att en etikett kan, men behöver inte nödvändigtvis, finnas på raden.	
L: <code>.space n{,v}</code>	reserverar n konsekutiva bytes vars initialvärden är v . Om v utelämnas sätts initialvärdet till 0.
L: <code>.byte n1,n2..</code>	Avsätter i följd i minnet en byte för varje argument. Respektive byte ges konstantvärdet $n1, n2$ etc. Följden placeras med början på adress L.
L: <code>.hword n1,n2..</code>	Avsätter i följd i minnet ett 16 bitars ord för varje argument. Respektive ord ges konstantvärdet $n1, n2$ etc. Följden placeras med början på adress L.
L: <code>.word n1,n2..</code>	Avsätter i följd i minnet ett 32 bitars ord för varje argument. Respektive ord ges konstantvärdet $n1, n2$ etc. Följden placeras med början på adress L.
L: <code>.dword n1,n2..</code>	Avsätter i följd i minnet ett 64 bitars ord för varje argument. Respektive ord ges konstantvärdet $n1, n2$ etc. Följden placeras med början på adress L.
<code>.align (n log 2)</code>	Garanterar att påföljande adress är jämnt delbar med 2^n
L: <code>.ascii "..."</code>	Avsätter en textsträng i minnet.
L: <code>.float fc1,fc2,..</code>	Avsätter en flyttalskonstant (32 bitar) för varje argument, i minnet. Respektive ord ges konstantvärdet $fc1, fc2$ etc. Följden placeras med början på adress L.
L: <code>.double fd1,fd2,..</code>	Avsätter en flyttalskonstant (64 bitar) för varje argument, i minnet. Respektive ord ges konstantvärdet $fd1, fd2$ etc. Följden placeras med början på adress L.
L: <code>.org n</code>	Följande kod, data placeras på offset n , från början av aktuell sektion.
<code>.equ sym, val</code>	Definiera symbolen sym med värdet val .
<code>.section .text</code> <code>.section .data</code> <code>.section namn</code>	Dessa sektioner innehåller själva programmet. Länkaren behandlar dem som separata men likvärdiga delar. Allt du kan säga om en sektion gäller även för en annan. När programmet körs är det dock vanligt att <code>text</code> -sektionen är konstant, dvs. ingen möjlighet till självmodifierande kod. <code>Text</code> -sektionen kan delas mellan olika program (<i>processer</i>): det innehåller instruktioner, konstanter och liknande. <code>data</code> -sektionen i ett program som körs är vanligtvis modifierbart: till exempel placeras initierade C-variabler i <code>data</code> -avsnittet. Sektioner kan ha godtyckliga namn så länge dessa uppfyller kraven för en
<code>.section .bss</code>	Denna sektion innehåller nollställda byte när programmet startas. Den används för oinitierade variabler med permanent lagring. Längden på varje delprograms <code>bss</code> -sektion är viktig, men eftersom den bara innehåller nollställda byte finns det inget behov av att lagra explicita nollbyte i objektfilen. Nollställningen görs i stället från <i>run-time biblioteket (startup)</i> innan <code>main</code> startas. <code>bss</code> -sektionen infördes för att eliminera explicita nollor från objektfiler och spara lagringsutrymmen.
<code>.section .absolute</code>	Adress 0 i denna sektion motsvarar alltid runtime-adress 0. Detta är användbart då man vill referera till en adress som länkaren inte får ändra vid relokering. I denna mening talar vi om absoluta adresser som "icke-relokerbara", dvs. de ändras inte under länkningen.

Exempel:

```
.space 4 # reservera 4 bytes, med initialt innehåll 0
.space 2,0xFF # reservera två bytes med initialt innehåll 0xFF

c: .byte 'A' # Skapa symbolen 'c' och tilldela värdet 'A'
s: .hword 1000 # Skapa symbolen 's' och tilldela värdet 1000
i: .word 100000 # Skapa symbolen 's' och tilldela värdet 1000
.ascii "String literal\0" # Textsträng med avslutande tecken för strängslut

.align 1 # påföljande adress garanteras vara jämnt delbar med 2
.align 2 # påföljande adress garanteras vara jämnt delbar med 4
.align 3 # påföljande adress garanteras vara jämnt delbar med 8

.equ dataport,0xF000 # konstant symbol 'dataport' med värdet 0xF000
```


mtvec (Exception base register)

Register `mtvec` innehåller två fält med konfigureringen för hur undantagshantering ska startas.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	mnemonic	
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	MODE
BASE																	MODE										mtvec						

MODE = 0 (00): PC <- BASE "RISC-V uniform"

MODE = 1 (01): PC <- BASE+(vektornummer×4) "RISC-V vektoriserad"

MODE = 2 (10): Reserverad, får ej användas

MODE = 3 (11): PC <- M [PC <- BASE+(vektornummer×4)] "WCH-specifik"

Vektorarean (*trap vector area*) är avsedd att innehålla adresser till hanteringsrutiner (*exception functions*) för de olika undantag som kan uppträda. Varje undantag karakteriseras av en identifikation (Vektornummer) som avgör varifrån (i vektorarean) hanteringen ska startas.

Vektor nummer	Prioritet	Prioritetstyp	Namn	Beskrivning	Adress
0	-	-	-	RESET	0x000000
1	-	-	-	Reserverad	0x000004
2	-5	fix	NMI	Icke maskerbart avbrott	0x000008
3	-4	fix	Hard fault	Undantagskoder: 0,1,2,4,5,6,7 (se Error! Reference source not found. nedan)	0x00000C
4	-	-	-	Reserverad	0x000010
5	-3	fix	Ecall-M	Ecall trap i M-läge	0x000014
6	-	-	-	Reserverad	0x000018
7	-	-	-	Reserverad	0x00001C
8	-2	fix	Ecall-U	Ecall trap i U-läge	0x000020
9	-1	fix	Breakpoint	Brytpunkt trap	0x000024
10	-	-	-	Reserverad	0x000028
11	-	-	-	Reserverad	0x00002C
12	0	0-15	SysTick	Räkarkrets	0x000030
13	-	-	-	Reserverad	0x000034
14	1	0-15	SWI	Software interrupt	0x000038
15	-	-	-	Reserverad	0x00003C
16-255			Interrupt	Avbrott från periferienheter	

mepc (Exception program counter)

Vid undantag sätts `mepc` till adressen där undantaget upptäcks (eller adressen till den omedelbart påföljande instruktionen) och ger då information om var exekveringen kan återupptas efter undantagshantering. Instruktionen `mret` återställer programräknaren från `mepc`. Observera att, beroende på undantag, kan `mepc` behöva modifieras av hanteringsrutinen.

Undantag	mepc
Avbrott	Adress till instruktionen omedelbart efter den instruktion som exekveras i det ögonblick avbrottet upptäckts,
Trap	Adress till instruktionen som orsakar undantaget

mtval (Exception value register)

Registret `mtval` är avsett att hålla kompletterande information baserat på vilket undantag som inträffat.

- Om undantaget orsakas av en brytpunkt lagras adressen till brytpunkten.
- Om undantaget orsakas av minnesåtkomst lagras adressen för minnesåtkomsten. Exempelvis adressfel, otillåten instruktionsadress, ej tillräcklig behörighet
- Om undantaget orsakas av en otillåten instruktion lagras operationskoden för instruktionen. Exempelvis icke existerande operationskod, instruktionen `unimp`, försök att läsa/skriva icke existerande CSR.
- För övriga undantag sätts `mtval` till 0

gintennr (Global interrupt enable register)

Registret `mstatus` medger bara åtkomst i M-läge. Innehållet i detta register är en instans av `mstatus` där bitarna kan ändras även då processorn är i U-läge. Detta tillåter alltså även program i U-läge att temporärt maskera avbrott för att exempelvis skapa så kallade *kritiska regioner* i programmet. *NMI* och *traps* påverkas inte utan hanteras normalt.

mcause (Exception cause register)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register	
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	mcause
EXCEPTION CODE																																	

Vid ett undantag skrivs en kod, som specificerar orsaken till undantaget, till register `mcause` av hårdvaran. I övrigt kan bara registret ändras av program med en CSR-instruktion.

Bit 31 i registret sätts baserat på undantagets natur, se tabell.

- *Synkron*: orsaken kan härledas till en punkt i programmet och är dessutom repeterbar.
- *Asynkron*: upphovet är en händelse oberoende av programmets exekvering, dvs. någon yttre händelse och sällan fullständigt repeterbar.
- *Precis*: instruktionen kan slutföras i sin helhet och all information om undantaget har fångats.
- *Imprecis*: instruktionen kan inte fullföljas på grund av att fel uppstod under någon fas av instruktionen.

mcause[31]	Undantagskod	Typ	Undantag
1	0-1	-	Reserverat
1	2	precis/asynkron	NMI
1	3-11	-	Reserverat
1	12	precis/ asynkron	SysTick
1	13	-	Reserverat
1	14	precis/synkron	SWI
1	15	-	Reserverat
1	16-255	precis/ asynkron	Externa avbrott
0	0	synkron	Otillåten instruktionsadress
0	1	synkron	Fetch command access error
0	2	synkron	Otillåten instruktion
0	3	synkron	Brytpunkt
0	4	synkron	Load instruction access address misalignment
0	5	imprecis/asynkron	Load command access error
0	6	synkron	Store/AMO instruction address misalignment
0	7	imprecis/asynkron	Store/AMO command access error
0	8	synkron	E-call U
0	11	synkron	E-call M

PFIC, Programmable Fast Interrupt Controller

För varje enskilt avbrott kontrollerat av PFIC finns följande funktioner:

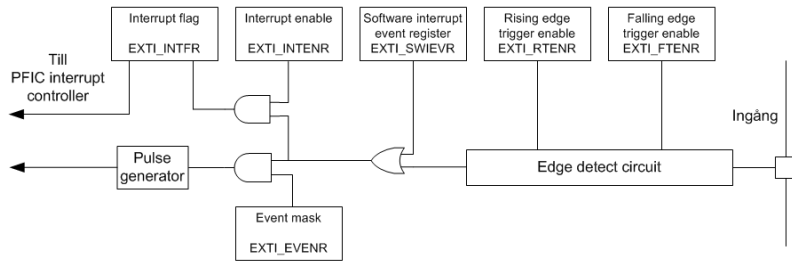
- *Interrupt Enabled Status*, PFIC_ISRx
- *Interrupt Pending*, PFIC_IPRx
- *Interrupt Set Enable*, PFIC_IENRx
- *Interrupt Clear Enable*, PFIC_IRERx
- *Interrupt Set Pending*, PFIC_IPSRx
- *Interrupt Clear Pending*, PFIC_IPRRx
- *Interrupt Activation*, PFIC_IACTR

PFIC_ISRx	x=1..4, Interrupt Enable Status Register. För varje avbrottsnummer anger här en bit om avbrott tillåts (is enabled).
PFIC_IPRx	x=1..4, Interrupt Pending status Register. För varje avbrottsnummer anger här en bit om avbrott avvaktar (is pending).
PFIC_IHRESDR	Interrupt Priority Threshold Configuration Register
PFIC_CFGR	Interrupt Configuration Register, se användarhandbok
PFIC_GISR	Interrupt Global Status Register, anger om något avbrott är avvaktande eller aktivt, anger nivå vid nästlade avbrott
PFIC_VTFIDR	Interrupt ID configuration Register, se användarhandbok
PFIC_VTFADDRx	x=0..3 Interrupt Address Register X, se användarhandbok
PFIC_IENRx	x=1..4, Interrupt Enable set Register. För varje avbrottsnummer anger här en bit att avbrott tillåts (set enable).
PFIC_IRERx	x=1..4, Interrupt Enable reset Register. För varje avbrottsnummer anger här en bit att avbrott tillåts (set enable).
PFIC_IPSRx	x=1..4, Interrupt Pending Set Register. För varje avbrottsnummer anger här en bit att avbrott sätts avvaktande (set pending).
PFIC_IPRRx	x=1..4, Interrupt Pending Reset Register. För varje avbrottsnummer anger här en bit att avbrott sätts ej avvaktande (clear pending).
PFIC_IACTRx	x=1..4, Interrupt Activation Register. För varje avbrottsnummer anger här en bit att avbrott aktiveras (set active).
PFIC_IPRIORx	x=0..63, Interrupt Priority Configuration Register. Modulen stöder upp till 256 avbrott där varje avbrottsnummer har ett 8-bitars prioritetsfält.
PFIC_SCTLR	System Control Register, se användarhandbok.

offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	mnemonic	
0xE000E000	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT ENABLED STATUS[31:0]	PFIC_ISR1
0xE000E004	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT ENABLED STATUS[63:32]	PFIC_ISR2	
0xE000E008	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT ENABLED STATUS[95:64]	PFIC_ISR3	
0xE000E00C	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT ENABLED STATUS[103:96]	PFIC_ISR4	
0xE000E020	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT PENDING[31:0]	PFIC_IPR1	
0xE000E024	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT PENDING [63:32]	PFIC_IPR2	
0xE000E028	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT PENDING [95:64]	PFIC_IPR3	
0xE000E02C	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT PENDING [103:96]	PFIC_IPR4	
0xE000E100	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT ENABLE SET[31:0]	PFIC_IENR1	
0xE000E104	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT ENABLE SET[63:32]	PFIC_IENR2	
0xE000E108	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT ENABLE SET[95:64]	PFIC_IENR3	
0xE000E10C	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT ENABLE SET[103:96]	PFIC_IENR4	
0xE000E180	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT ENABLE RESET[31:0]	PFIC_IRER1	
0xE000E184	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT ENABLE RESE [63:32]	PFIC_IRER2	
0xE000E188	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT ENABLE RESE [95:64]	PFIC_IRER3	
0xE000E18C	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT ENABLE RESE[103:96]	PFIC_IRER4	
0xE000E200	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT PENDING SET[31:0]	PFIC_IPSR1	
0xE000E204	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT PENDING SET [63:32]	PFIC_IPSR2	
0xE000E208	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT PENDING SET [95:64]	PFIC_IPSR3	
0xE000E20C	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT PENDING SET [103:96]	PFIC_IPSR4	
0xE000E280	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT PENDING RESET[31:0]	PFIC_IPRR1	
0xE000E284	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT PENDING RESE [63:32]	PFIC_IPRR2	
0xE000E288	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT PENDING RESE [95:64]	PFIC_IPRR3	
0xE000E28C	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT PENDING RESE [103:96]	PFIC_IPRR4	
0xE000E300	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT ACTIVATION[31:0]	PFIC_IACTR1	
0xE000E304	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT ACTIVATION [63:32]	PFIC_IACTR2	
0xE000E308	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT ACTIVATION [95:64]	PFIC_IACTR3	
0xE000E30C	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	INTERRUPT ACTIVATION [103:96]	PFIC_IACTR4	

EXTI External Interrupt

Adress	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register	
0x40010400																																		EXTI_INTENR
0x40010404																																		EXTI_EVENR
0x40010408																																		EXTI_RTENR
0x4001040C																																		EXTI_FTENR
0x40010410																																		EXTI_SWIEVR
0x40010414																																		EXTI_INTFR



EXTI_INTENR Interrupt enable register <table border="1"> <thead> <tr> <th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th> <th>mnemonic</th> </tr> </thead> <tbody> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> <td>EXTI_INTENR</td> </tr> </tbody> </table>	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	mnemonic																																		EXTI_INTENR	Bit INTENR[22..0]: Avbrottsmask för avbrottslina x. 0: Avbrott är fränslaget (ej tillåtet) 1: Avbrott är tillslaget (tillåtet)
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	mnemonic																																				
																																	EXTI_INTENR																																			
EXTI_EVENR Event enable register <table border="1"> <thead> <tr> <th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th> <th>mnemonic</th> </tr> </thead> <tbody> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> <td>EXTI_EVENR</td> </tr> </tbody> </table>	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	mnemonic																																		EXTI_EVENR	Bit EVENR[22..0]: Eventmask för avbrottslina x. 0: Event är fränslaget (ej tillåtet) 1: Event är tillslaget (tillåtet)
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	mnemonic																																				
																																	EXTI_EVENR																																			
EXTI_RTENR Rising edge trigger enable register <table border="1"> <thead> <tr> <th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th> <th>mnemonic</th> </tr> </thead> <tbody> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> <td>EXTI_RTENR</td> </tr> </tbody> </table>	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	mnemonic																																		EXTI_RTENR	Bit RTENR[22..0]: Trigg på positiv flank för avbrottslina x. 0: Trigg på positiv flank genererar ej avbrott 1: Trigg på positiv flank genererar avbrott
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	mnemonic																																				
																																	EXTI_RTENR																																			
EXTI_FTENR Falling edge trigger enable register <table border="1"> <thead> <tr> <th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th> <th>mnemonic</th> </tr> </thead> <tbody> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> <td>EXTI_FTENR</td> </tr> </tbody> </table>	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	mnemonic																																		EXTI_FTENR	Bit FTENR[22..0]: Trigg på negativ flank för avbrottslina x. 0: Trigg på negativ flank genererar ej avbrott 1: Trigg på negativ flank genererar avbrott
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	mnemonic																																				
																																	EXTI_FTENR																																			
EXTI_SWIEVR Software interrupt event register <table border="1"> <thead> <tr> <th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th> <th>mnemonic</th> </tr> </thead> <tbody> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> <td>EXTI_SWIEVR</td> </tr> </tbody> </table>	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	mnemonic																																		EXTI_SWIEVR	Bit SWIER[22..0]: Om avbrott är aktiverat för lina x, kan programvara aktivera detta genom att skriva '1' till motsvarande bit i detta register. Denna bit måste sedan återställs av hanteringsrutinen.
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	mnemonic																																				
																																	EXTI_SWIEVR																																			
EXTI_INTFR Interrupt flag register <table border="1"> <thead> <tr> <th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th> <th>mnemonic</th> </tr> </thead> <tbody> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> <td>EXTI_INTFR</td> </tr> </tbody> </table>	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	mnemonic																																		EXTI_INTFR	Bit INTFR[22..0]: Motsvarande bit i detta register sätts till 1 av hårdvaran då ett triggvillkor är uppfyllt. Då hanteringsrutinen avslutas ska avbrottsbegäran kvitteras så att biten återställs. Registret är därför även skrivbart och biten återställs genom att skrivas med '1'. 0: Ingen trigg. 1: Trigg har uppträtt, avbrott avvaktar (<i>is pending</i>)
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	mnemonic																																				
																																	EXTI_INTFR																																			

AFIO EXTI Control

offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register	
0x40010000																																		AFIO_ECR
0x40010004																																		AFIO_PCFR1
0x40010008																																		AFIO_EXTICR1
0x4001000C																																		AFIO_EXTICR2
0x40010010																																		AFIO_EXTICR3
0x40010014																																		AFIO_EXTICR4
0x4001001C																																		AFIO_PCFR2

0000: PA[x] 0001: PB[x] 0010: PC[x] 0011: PD[x] 0100: PE[x]

address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
0x40010008	EXTI3[3:0]			EXTI2[3:0]			EXTI1[3:0]			EXTI0[3:0]						AFIO_EXTICR1	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
0x4001000C	EXTI7[3:0]			EXTI6[3:0]			EXTI5[3:0]			EXTI4[3:0]						AFIO_EXTICR2	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
0x40010010	EXTI11[3:0]			EXTI10[3:0]			EXTI9[3:0]			EXTI8[3:0]						AFIO_EXTICR3	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
0x40010014	EXTI15[3:0]			EXTI14[3:0]			EXTI13[3:0]			EXTI12[3:0]						AFIO_EXTICR4	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Vektortabell

Prioriteter 0 och större är programmerbara.

Id	priority	name	description	vector offset
-	-			0x0000 0000
-	-			0x0000 0004
2	-5	NMI	Non maskable interrupt.	0x0000 0008
3	-4	HardFault		0x0000 000C
4	-		Reserved	0x0000 0010
5	-3	Ecall-M	Callback interrupt in machine mode	0x0000 0014
6-7	-		Reserved	0x0000 0018-0x0000 001C
8	-2	Ecall-U	Callback interrupt in user mode	0x0000 0020
9	-1	Breakpoint	Breakpoint callback interrupt	0x0000 0024
10-11	-		Reserved	0x0000 0028-0x0000 002C
12	0	SysTick	System timer interrupt	0x0000 0030
13	-		Reserved	0x0000 0034
14	1	SWI	Software interrupt	0x0000 0038
15	-		Reserved	0x0000 003C
16	2	WWDG	Window Watchdog interrupt	0x0000 0040
17	3	PVD	Power voltage detector interrupt (EXTI)	0x0000 0044
18	4	TAMPER	Tamper interrupt	0x0000 0048
19	5	RTC	Real-time clock interrupt	0x0000 004C
20	6	FLASH	Flash memory global interrupt	0x0000 0050
21	7	RCC	RCC and clock control interrupt	0x0000 0054
22	8	EXTI0	EXTI Line0 interrupt	0x0000 0058
23	9	EXTI1	EXTI Line1 interrupt	0x0000 005C
24	10	EXTI2	EXTI Line2 interrupt	0x0000 0060
25	11	EXTI3	EXTI Line3 interrupt	0x0000 0064
26	12	EXTI4	EXTI Line4 interrupt	0x0000 0068
27	13	DMA1_CH1	DMA1 channel1 global interrupt	0x0000 006C
28	14	DMA1_CH2	DMA1 channel2 global interrupt	0x0000 0070
29	15	DMA1_CH3	DMA1 channel3 global interrupt	0x0000 0074
30	16	DMA1_CH4	DMA1 channel4 global interrupt	0x0000 0078
31	17	DMA1_CH5	DMA1 channel5 global interrupt	0x0000 007C
32	18	DMA1_CH6	DMA1 channel6 global interrupt	0x0000 0080
33	19	DMA1_CH7	DMA1 channel7 global interrupt	0x0000 0084
34	20	ADC1_2	ADC1 and ADC2 global interrupts	0x0000 0088
35	21	USB_HP or CAN1_TX	USB_HP or CAN1_TX interrupts	0x0000 008C
36	22	USB_LP or CAN1_RX0	USB_LP or CAN1_RX0 interrupts	0x0000 0090
37	23	CAN1_RX1	CAN1_RX1 interrupt	0x0000 0094
38	24	CAN1_SCE	CAN1_SCE interrupt	0x0000 0098
39	25	EXTI9_5	Line[9:5] interrupts	0x0000 009C
40	26	TIM1_BRK	TIM1 break interrupt	0x0000 00A0
41	27	TIM1_UP	TIM1 update interrupt	0x0000 00A4
42	28	TIM1_TRG_COM	TIM1 trigger and commutation interrupts	0x0000 00A8
43	29	TIM1_CC	TIM1 capture compare interrupt	0x0000 00AC
44	30	TIM2	TIM2 global interrupt	0x0000 00B0
45	31	TIM3	TIM3 global interrupt	0x0000 00B4
46	32	TIM4	TIM4 global interrupt	0x0000 00B8
47	33	I2C1_EV	I2C1 event interrupt	0x0000 00BC
48	34	I2C1_ER	I2C1 error interrupt	0x0000 00C0
49	35	I2C2_EV	I2C2 event interrupt	0x0000 00C4
50	36	I2C2_ER	I2C2 error interrupt	0x0000 00C8
51	37	SPI1	SPI1 global interrupt	0x0000 00CC
52	38	SPI2	SPI2 global interrupt	0x0000 00D0
53	39	USART1	USART1 global interrupt	0x0000 00D4
54	40	USART2	USART2 global interrupt	0x0000 00D8
55	41	USART3	USART3 global interrupt	0x0000 00DC
56	42	EXTI15_10	EXTI Line[15:10] interrupts	0x0000 00E0
57	43	RTC Alarm	RTC alarm interrupt (EXTI)	0x0000 00E4
58	44	USBWakeUp	USB wakeup interrupt (EXTI)	0x0000 00E8
59	45	TIM8_BRK	TIM8 break interrupt	0x0000 00EC
60	46	TIM8_UP	TIM8 update interrupt	0x0000 00F0
61	47	TIM8_TRG_COM	TIM8 trigger and commutation interrupts	0x0000 00F4
62	48	TIM8_CC	TIM8 capture Compare interrupt	0x0000 00F8
63	49	RNG	RNG global interrupt	0x0000 00FC
64	50	-	Reserved	0x0000 0100
65	51	SDIO	SDIO global interrupt	0x0000 0104
66	52	TIM5	TIM5 global interrupt	0x0000 0108
67	53	SPI3	SPI3 global interrupt	0x0000 010C
68	54	USART4	USART4 global interrupt	0x0000 0110
69	55	USART5	USART5 global interrupt	0x0000 0114
70	56	TIM6	TIM6 global interrupt,	0x0000 0118
71	57	TIM7	TIM7 global interrupt	0x0000 011C
72	58	DMA2_CH1	DMA2 Channel 1 global interrupt	0x0000 0120
73	59	DMA2_CH2	DMA2 Channel 2 global interrupt	0x0000 0124
74	60	DMA2_CH3	DMA2 Channel 3 global interrupt	0x0000 0128
75	61	DMA2_CH4	DMA2 Channel 4 global interrupt	0x0000 012C
76	62	DMA2_CH5	DMA2 Channel 5 global interrupt	0x0000 0130
77	63	ETH	Ethernet global interrupt	0x0000 0134
78	64	ETH_WKUP	ETH wakeup interrupt	0x0000 0138
79	65	CAN2_TX	CAN2_TX global interrupt	0x0000 013C
80	66	CAN2_RX0	CAN2_RX0 global interrupt	0x0000 0140
81	67	CAN2_RX1	CAN2_RX1 global interrupt	0x0000 0144
82	68	CAN2_SCE	CAN2_SCE global interrupt	0x0000 0148
83	69	OTG_FS	Full-speed OTG interrupt	0x0000 014C
84	70	USBHSWakeUp	High-speed USB wakeup interrupt	0x0000 0150

Quick Guide – får användas under tentamen i "Maskinorienterad programmering"

85	71	USBHS	High-speed USB global interrupt	0x0000 0154
86	72	DVP	DVP global interrupt	0x0000 0158
87	73	USART6	USART6 global interrupt	0x0000 015C
88	74	USART7	USART7 global interrupt	0x0000 0160
89	75	USART8	USART8 global interrupt	0x0000 0164
90	76	TIM9 BRK	TIM9 break interrupt	0x0000 0168
91	77	TIM9 UP	TIM9 update interrupt	0x0000 016C
92	78	TIM9 TRG COM	TIM9 trigger and communication interrupts	0x0000 0170
93	79	TIM9 CC	TIM9 capture compare interrupt	0x0000 0174
94	80	TIM10 BRK	TIM10 break interrupt	0x0000 0178
95	81	TIM10 UP	TIM10 update interrupt	0x0000 017C
96	82	TIM10 TRG COM	TIM10 trigger and communication interrupts	0x0000 0180
97	83	TIM10 CC	TIM10 capture compare interrupt	0x0000 0184
98	84	DMA2 CH6	DMA2 channel6 global interrupt	0x0000 0188
99	85	DMA2 CH7	DMA2 channel7 global interrupt	0x0000 018C
100	86	DMA2 CH8	DMA2 channel8 global interrupt	0x0000 0190
101	87	DMA2 CH9	DMA2 channel9 global interrupt	0x0000 0194
102	88	DMA2 CH10	DMA2 channel10 global interrupt	0x0000 0198
103	89	DMA2 CH11	DMA2 channel11 global interrupt	0x0000 019C

SYSTEMKRETSAR OCH PERIFERIEHETER

Räknarkrets SYSTICK

Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
0xE000F000																																	STK_CTLR
0xE000F004																																	STK_SR
0xE000F008																																	STK_CNTL
0xE000F00C																																	STK_CNTH
0xE000F010																																	STK_CMPLR
0xE000F014																																	STK_CMPHR

STK_CTLR Styrregister

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	STK_CTLR	
RW																										W1	RW	RW	RW	RW	RW	RW	
SWIE																											INIT	MODE	STRE	STCLK	STIE	STE	

Bit 31 SWIE: Software Interrupt Flag Enable

SWI aktiveras genom att 1 skrivs till biten, återställs genom att 0 skrivs till biten. Vi återkommer till detta i kapitlet om undantagshantering.

Bit 5 INIT: Counter Register Load Value

Räknarregistret initieras (se bit 4, MODE).

1: CNTL/CNTH nollställs om MODE är 0 ("upcount")

CNTL/CNTH laddas med värden från CMPLR/CMPHR om MODE är 1

("downcount")

0: Ingen initiering av räknarregistret.

Bit 4 MODE: Count Mode

1: Nedräkning, räknarens slutvärde är 0

0: Uppräkning, räknarens slutvärde är CMPLR/CMPHR

Bit 3 STRE: Reload Enable

1: Då räknaren nått ett slutvärde laddas på nytt CNTL och CNTH, beroende på biten MODE

0: Räknaren fortsätter att räkna, beroende på biten MODE

Bit 2 STCLK: Timer Clock Source

1: Räknarens tidbas är systemklockan (max 144 MHz)

0: Räknarens tidbas är systemklockan dividerat med 8

Bit 1 STIE: Timer Interrupt Enable

Avbrott kan genereras då räknaren nått sitt slutvärde. Vi återkommer till detta i kapitlet om undantagshantering.

1: Avbrott genereras

0: Inget avbrott genereras

Bit 0 STE: Timer Enable

Aktivera räknaren.

1: Räknarens värde klockas och räknar vid klockpulserna

0: Räknarens värde påverkas ej av klockan

STK_SR Statusregister

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	STK_SR
																																RW
																																CNTIF

Bit 0 CNTIF: Count Value Compare Flag

Vid läsning anger biten om räknaren nått slutvärdet, dvs. CMPLR/CMPHR om arbetssättet är uppräkning, respektive 0 om arbetssättet är nedräkning.

1: Räknarvärdet har nått slutvärdet.

0: Räknarvärdet har ännu ej nått slutvärdet.

Återställning (till 0) sker genom att 0 skrivs till biten. En skrivning 1 ignoreras (W0).

STK_CNTL Counter Low Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	STK_CNTL
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW		
CNTL[31..0]																																

Innehåller de 32 minst signifikanta bitarna av räknarens momentanvärde.

STK_CNTH Counter High Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	STK_CNTH
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW		
CNTH[63..32]																																

Innehåller de 32 mest signifikanta bitarna av räknarens momentanvärde.

STK_CMPLR Counter Compare Low Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	STK_CMPLR
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW		
CMPHR[31..0]																																

Innehåller de 32 minst signifikanta bitarna av räknarens maximala värde.

STK_CMPHR Counter Compare High Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	STK_CMPHR
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW		
CMPHR[63..32]																																

Innehåller de 32 mest signifikanta bitarna av räknarens maximala värde.

GPIO

General Purpose Input Output

GPIO A: 0x40010800

GPIO B: 0x40010C00

GPIO C: 0x40011000

GPIO D: 0x40011400

GPIO E: 0x40011800

offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
0																																	GPIO_CFGLR
4																																	GPIO_CFGHR
8																																	GPIO_INDR
0xC																																	GPIO_OUTDR
0x10																																	GPIO_BSHR
0x14																																	GPIO_BCR
0x18																																	GPIO_LCKR

CFGLR, CFGHR Port configuration registers

offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	mnemonic
0x0	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	GPIO_CFGLR
																																	pin7 pin6 pin5 pin4 pin3 pin2 pin1 pin0

offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	mnemonic
0x4	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	GPIO_CFGHR
																																	pin15 pin14 pin13 pin12 pin11 pin10 pin9 pin8

pinX		function	
CNF	MODE		
0	0	0	Analog input mode
0	1	0	Floating input mode
1	0	0	Digital input mode with pull up/pull down
1	1	0	DO NOT USE
0	0	1	Digital output, push/pull – max 10MHz
0	0	1	Digital output, push/pull – max 2MHz
0	0	1	Digital output, push/pull – max 50MHz
0	1	0	Digital output, open drain – max 10MHz
0	1	0	Digital output, open drain – max 2MHz
0	1	1	Digital output, open drain – max 50MHz
1	0	0	Alternate function push/pull – max 10MHz
1	0	1	Alternate function push/pull – max 2MHz
1	0	1	Alternate function push/pull – max 50MHz
1	1	0	Alternate function open drain – max 10MHz
1	1	0	Alternate function open drain – max 2MHz
1	1	1	Alternate function open drain – max 50MHz

GPIO_OUTDR Output Data Register

offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	mnemonic
0xC																	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	GPIO_OUTDR

Då pinnen programmerats som en utgång bestäms pinnens nivå via detta register. Registret skrivs med 16-bitars format. Läsning från registret returnerar det senast skrivna värdet.

Då pinnen programmerats som ingång, pull-up/pull-down, anger motsvarande bit i stället om portpinnen ska vara pull-up (1) eller pull-down (0).

GPIO_INDR Input Data Register

Registret läses med 16-bitars format. Skrivning har ingen effekt

offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	mnemonic
8																	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	GPIO_INDR

GPIO_BSHR Bit Set Reset Register

Registret skrivs med 16 (enbart *bit set*) eller 32 bitar.

offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	mnemonic
0x10	w1	w1	w1	w1	w1	w1	w1	w1	w1	w1	w1	w1	w1	w1	w1	w1	GPIO_BSHR
	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	w1	w1	w1	w1	w1	w1	w1	w1	w1	w1	w1	w1	w1	w1	w1	w1	
	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0	

GPIO_BCR Bit Clear Register

offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	mnemonic
0x14																	GPIO_BCR
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	w1	w1	w1	w1	w1	w1	w1	w1	w1	w1	w1	w1	w1	w1	w1	w1	
	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0	

GPIO_LCKR configuration LoCK Register

offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	mnemonic
0x1C																LCKK	GPIO_LCKR
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0	

Biten LCKK är 0 då konfigurationsregistren är upplåsta och slås om till 1, då läsekvensen (write 1/write0/write1) utförts.

TIMER6/TIMER7

TIM6: 0x4000 1000 - 0x4000 13FF

TIM7: 0x4000 1400 - 0x4000 17FF

offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register	
0																																		TIMx_CTLR1
4																																		
8																																		
0xC																																		TIMx_DMAINTENR
0x10																																		TIMx_INTFR
0x14																																		TIM6_SWEVGR
0x18																																		
0x1C																																		
0x20																																		
0x24																																		TIMx_CNT
0x28																																		TIMx_PSC
0x2C																																		TIMx_ATRLR

TIMxPSC och TIMx_ATRLR anger räknarkretsens tidsbas, dvs. räknarintervall till *update event*. Räknarkretsarna använder samma klocka som tidsbas, antalet klockpulser under 1 sekund, dvs frekvensen för UE, bestäms av:

$$UE \text{ Hz} = \frac{CLK \text{ Hz}}{(PSC + 1)(ATRLR + 1)}$$

Räknarens klockfrekvens (CLK) beror på hur systemet initierats. För MD307 med dbg307 gäller 144 MHz.

TIM6 och TIM7 styrregister CTLR1 (Control Register 1)

offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
0x00									ARPE				OPM	URS	UDIS	CEN	TIMxCTLR1

Bit 7 ARPE: Auto-reload preload enable

Då register TIMx_ATLR uppdateras med nytt värde kan detta överföras omedelbart, eller efter nästa kompletta räknarintervall är klart.

0: TIMx_ATLR register ändras via ett buffertregister.

1: TIMx_ATLR register ändras omedelbart.

Bit 3 OPM: One-pulse mode

0: Räknaren fortsätter efter ett räknarintervall (*kontinuerligt arbetssätt*)

1: Räknaren stoppas efter ett räknarintervall genom att CEN-biten nollställs.

Bit 2 URS: Update request source

Biten kontrollerar källorna till UEV (*update event*).

0: Någon av följande händelser kan generera *update interrupt* eller DMA-begäran.

– Räknaren når max eller min-värde. (*overflow* eller *underflow*)

– UG-biten i EGR (se nedan) sätts till 1 av programvara.

– *Update generation* från en slav-räknare

1: Endast räknare *overflow/underflow* genererar *update interrupt* eller DMA-begäran

Bit 1 UDIS: Update disable

Biten bestämmer om UEV (*update event*) kan genereras.

0: UEV är möjligt och kan genereras. Se även URS. Register ges begynnelsevärden från buffertar.

1: UEV är avstängd, inget UE (*update event*) genereras. Skuggregister (hos buffrade register ARR och PCS) uppdateras bara då UG-biten sätts till 1.

Bit 0 CEN: Counter enable

0: Räknarkretsen är deaktiverad

1: Räknarkretsen är aktiverad

TIM6 och TIM7 styrregister DMAINTENR (DMA/interrupt enable register)

offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
0x0C									UDE							UIE	TIMxDMAINTENR

Bit 8 UDE: Update DMA request enable

0: *Update* vid DMA-begäran deaktiverad.

1: *Update* vid DMA-begäran aktiverad.

Bit 0 UIE: Update interrupt enable

0: *Update* genererar inget avbrott.

1: *Update* genererar avbrott.

TIM6 och TIM7 statusregister INTFR

offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
0x10																UIF	TIMxINTFR

Bit 0 UIF: Update Interrupt Flag

0: *update event* har ej inträffat

1: *update event* har inträffat

TIM6 och TIM7 styrregister SWEVGR (Event Generation Register)

offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
0x14																UG	TIMxSWEVGR

Bit 0 UG: Update generation

Ett program kan fås att generera en *update*-händelse. Typiskt används det för teständamål men andra tillämpningar är naturligtvis också möjliga. Biten kan sättas av programvara, återställs sedan av hårdvara.

0: Ingen åtgärd.

1: Genererar en *update*-händelse på samma sätt som om den hade genererats av den autonoma räknaren.

TIMx_CNT innehåller det aktuella räknarvärdet. Registret kan bara läsas.

PSC och ATRLR ger tillsammans räknarintervallet, dvs. maximala antalet pulser som räknas vilket då ger periodtiden till ett *update event*. CNT innehåller det aktuella räknarvärdet.

TIM6 och TIM7 räknare CNT (Counter)

offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
0x24																	TIMxCNT

TIM6 och TIM7 räknare PSC (Prescaler)

offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
0x28																	TIMxPSC

TIM6 och TIM7 räknare ATRLR (Auto Reload Register)

offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
0x2C																	TIMxATLR

USART

Universal synchronous asynchronous receiver transmitter

USART1: 0x40013800 UART5: 0x40005000

USART2: 0x40004400 UART6: 0x40001800

USART3: 0x40004800 UART7: 0x40001C00

UART4: 0x40004C00 UART8: 0x40002000

offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
0																																	USART_STATR
4																																	USART_DATAR
8																																	USART_BRR
0xC																																	USART_CTLR1
0x10																																	USART_CTLR2
0x14																																	USART_CTLR3
0x18																																	USART_GPR

USART_STATR Statusregister (vid RESET 0x00C0)

offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
							RW0	RW0	R	RW0	RW0	R	R	R	R	R	
0							CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE	USART_STATR

Bit 9 CTS: Clear To Send

Denna bit sätts av hårdvara om ETSC=1 när nivån på NCTS ingången ändras. Den återställs av programvara (genom att skriva 0 till biten). Ett avbrott genereras om CTSIE = 1 i registret UART_CR3.

0: Ingen förändring har skett på NCTS

1: En förändring har skett på NCTS

Bit 8 LBD: Lin Break Detected

Denna bit sätts av hårdvara när en så kallad *LIN break*-ram detekteras. Det återställs av programvara (genom att skriva 0 till biten). Ett avbrott genereras om LBDIE = 1 i registret USART_CR2.

0: LIN break-ram ej upptäckt

1: LIN break-ram avbrott upptäckt

Bit 7 TXE: Transmit data register empty

Denna bit sätts av hårdvara när innehållet av TDR registret har överförts till skiftregistret. Biten sätts till 0 vid skrivning till DATAR. Ett avbrott genereras om TXEIE i CTRL1 är 1 samtidigt som TXE är 1..

0: Dataregistrets sändardel är upptaget med en överföring.

1: Dataregistrets sändardel är klar att användas.

Bit 6 TC: Transmission Complete

Denna bit sätts då sändningen av en ram är komplett och om TXE =1. Ett avbrott genereras om TCIE i CTRL1 samtidigt är 1. TC nollställs av en läsning från STATR följt av en skrivning till DATAR eller genom att 0 skrivs till biten.

0: Överföring pågår

1: Överföringen är klar

Bit 5 RXNE: Receive data register not empty

Denna bit sätts då innehållet i skiftregister RDR har överförts till DATAR, dvs. ett nytt tecken har kommit. Ett avbrott genereras om RXNEIE i CTRL1 samtidigt är 1. Biten nollställs igen vid en läsning från DATAR. Biten kan också återställas genom att skriva en nolla till den.

0: Inget nytt innehåll i DATAR sedan senaste läsningen

1: Nytt innehåll finns i DATAR.

Bit 4 IDLE: Idle line detected

Denna bit sätts av hårdvaran när en tom ram, indikerandes att serieledningen är ledig, upptäcks. Ett avbrott genereras om IDLEIE CTRL1 samtidigt är 1. IDLE återställs av en läsning från STATR direkt följt av en läsning från DATAR.

0: Ingen tom ram har detekterats

1: En tom ram har detekterats

Bit 3 ORE: Overrun Error

Denna bit sätts av hårdvaran om ett nytt tecken anländer samtidigt som det finns ett oläst tecken i dataregistret ("overrun error"). Ett avbrott genereras om RXNEIE i CTRL1 samtidigt är 1. ORE återställs av en läsning från STATR följt av en läsning från DATAR.

0: Inget förlorat tecken

1: Mottaget tecken är överskrivet (förlorat)

1: Ramfel eller BREAK-ram detekterad

Bit 2 NE: Noise Error

Denna bit sätts av hårdvara när störningar i form av brus upptäcks i en mottagen ram. Biten återställs av en läsning från STATR följt av en läsning från DATAR.

0: Ingen störning

1: Störning detekterad

Bit 1 FE: Framing Error

Denna bit sätts av hårdvara när ett ramfel, oftast orsakat av förlorad synkronisering, upptäcks. Biten återställs av en läsning från STATR följt av en läsning från DATAR.

0: Inget ramfel

1: Ramfel detekterat

Bit 0 PE: Parity Error

Denna bit sätts av hårdvara när ett paritetsfel uppträder hos mottagaren. Biten återställs av en läsning från STATR följt av en läsning från DATAR. Programmet måste vänta på att RXNE-biten ettställts innan PE-biten återställs. Ett avbrott genereras om PEIE i CTRL1 samtidigt är 1..

0: Inget paritetsfel

1: Paritetsfel detekterat

USART_DATAR Dataregister

offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
							RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	
4							R9	R8	R7	R6	R5	R4	R3	R2	R1	R0	RDR
							T9	T8	T7	T6	T5	T4	T3	T2	T1	T0	TDR

Bit 8:0 DATAR [8:0]

Innehåller tecknet som tas emot eller sänds. DATAR har alltså en dubbel funktion (läs och skriv) eftersom det består av två olika fysiska register, ett för sändning (TDR) och en för mottagning (RDR). Vid sändning med paritet aktiverat (PCE bit satt till 1 i register CTRL1), är den mest signifikanta biten betydelselös eftersom den ersätts av pariteten för ordet. Vid mottagning med paritet aktiverat är det värde som avlästs i den mest signifikanta biten den mottagna pariteten..

USART_BRR Baudrate register (vid RESET: 0x0000)

offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
8	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	USART_BRR
	DIV_Mantissa[11:0]											DIV_Fraction[3:0]					

Baudraten för asynkron och SPI-mod bestäms av:

$$\text{Baudrate} = f_{\text{CK}} / (16 \times \text{USARTDIV})$$

USARTDIV består av DIV_Mantissa och DIV_Fraction

Bits 15:4 DIV_Mantissa [11: 0]: heltalsdelen av USARTDIV

Bits 3:0 DIV_Fraction [3: 0]: decimaldelen av USARTDIV.

USART_CR1 Control register 1 (vid RESET: 0x0000)

offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
0xC			RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	USART_CTLR1
			UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK	

Bit 13 UE: USART enable

0: USART deaktiverad

1: USART aktiverad

Bit 12 M: Word length

Biten bestämmer ordlängden.

0: 1 startbit, 8 databitar, 1 stoppbit

1: 1 start bit, 9 databitar, 1 stoppbit

Bit 11 WAKE: Wakeup method

Biten bestämmer uppväckningsmetoden.

0: Idle Line

1: Address Mark

Bit 10 PCE: Parity Control Enable

Biten bestämmer om paritetsbit ska kontrolleras. När paritetsbit används sätts den alltid in som MSB.

0: Ingen paritetsbit

1: Paritetsbit används

Bit 9 PS: Parity Selection

Med biten väljs typen av paritet.

0: Jämn paritet

1: Udda paritet

Bit 8 PEIE: PE Interrupt Enable

Biten sätts och nollställs av programvara.

0: Ingen funktion

1: USART avbrott genereras då PE=1 i STATR, dvs. paritetsfel.

Bit 7 TXEIE: TXE Interrupt Enable

Biten sätts och nollställs av programvara.

0: Ingen funktion

1: USART avbrott genereras då TXE=1 i STATR.

Bit 6 TCIE: TC Interrupt Enable

Biten sätts och nollställs av programvara.

0: Ingen funktion

1: USART avbrott genereras då TC =1 i STATR

Bit 5 RXNEIE: RXNE Interrupt Enable

Biten sätts och nollställs av programvara.

0: Ingen funktion

1: USART avbrott genereras då ORE=1 eller RXNE =1 i STATR

Bit 4 IDLEIE: IDLE Interrupt Enable

Biten sätts och nollställs av programvara.

0: Ingen funktion

1: USART avbrott genereras då IDLE =1 i STATR

Bit 3 TE: Transmitter Enable

Biten sätts och nollställs av programvara.

0: Sändare deaktiverad

1: Sändare aktiverad

Bit 2 RE: Receiver Enable

Biten sätts och nollställs av programvara.

0: Mottagaren är deaktiverad

1: Mottagaren aktiverad

Bit 1 RWU: Receiver wakeup

Biten bestämmer om mottagaren är i *mute*-mod eller inte. Biten sätts och nollställs av programvara och kan dessutom nollställas av hårdvaran då en *wakeup* sekvens uppträder hos mottagaren.

0: Mottagaren aktiv

1: Mottagare i *mute*-mod

Bit 0 SBK: Send break

Denna bit används för att sända BREAK. Biten sätts av programvara och återställs av hårdvaran då hela tecknet skickats.

0: Passiv

1: Skicka BREAK-tecken

U(S)ART_CR2 Control register 2 (vid RESET: 0x0000)

offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
0x10		RW		RW		RW	RW	RW	RW		RW	RW					USART_CTLR2
		LIN EN		STOP1[1:0]		CLK EN	CPOL	CPHA	LBCL		LBD IE	LBDL					ADD[3:0]

offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
0x10		RW		RW							RW	RW					UART_CTLR2
		LIN EN		STOP1[1:0]							LBD IE	LBDL					ADD[3:0]

Bit 15 Reserverad

Bit 14 LINEN: LIN mode enable

Biten sätts och nollställs av programvara.

0: LIN mode deaktiverad

1: LIN mode aktiverad

Med LIN-mod aktiverad kan USART skicka *LIN Synch Breaks* (13 låga bitar) via SBK-biten i CTLR1, och dessutom detektera *LIN Sync breaks* på bussen.

Bit 13..12 STOP: STOP bits

Dessa bitar används för att programmera antalet stoppbitar i protokollet.

00: 1 Stop bit

01: 0.5 Stop bit

10: 2 Stop bits

11: 1.5 Stop bit

Bit 11 CLKEN: Clock enable

0: SCLK deaktiverad

1: SCLK aktiverad

Bit 10 CPOL: Clock polarity

Biten bestämmer polariteten hos klockutgången SCLK i synkron mod. I kombination med CPHA-biten bestämmer denna bit relationen klocka/data hos signalen

0: SCLK är låg utanför sändarfönstret.

1: SCLK är hög utanför sändarfönstret.

Bit 9 CPHA: Clock phase

Biten bestämmer fasen hos klockutgången SCLK i synkron mod. I kombination med CPOL-biten bestämmer denna bit relationen klocka/data hos signalen

0: Första klocktransition låser data

1: Andra klocktransition låser data

Bit 8 LBCL: Last bit clock pulse

Denna bit avgör om även den sista databiten ska ha en associerad klockpuls.

0: Sista databiten har ingen klockpuls hos SCLK

1: Sista databiten har klockpuls hos SCLK

Not 1: Sista biten är den 8:e eller 9:e biten beroende på M i CTLR1.

Not 2: Kan inte användas med UART4 och UART5

Bit 6 LBDIE: LIN break detection interrupt enable

0: Inget avbrott genereras då LBD=1 i STATR

1: Avbrott genereras då LBD=1 i STATR

Bit 5 LBDL: LIN break detection length

0: 10-bit break detektering

1: 11-bit break detektering

Bits 3..0 ADD[3:0]: Address of the USART node

Bitfältet ger adressen till USART-noden i multiprocessor-applikationer under tyst mod.

Not: Bitarna CPOL, CPHA och LBCL får inte ändras då sändaren är aktiverad.

USART_CR3 Control register 3 (vid RESET: 0x0000)

offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
0x14						RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	USART_CTLR3
						CTS IE	CTSE	RTSE	DMAT	DMA R	SCEN	NACK	HDSSEL	IRLP	IREN	EIE	

Bit 10 CTSIE: CTS interrupt enable

0: Avbrott deaktiverat

1: Avbrott genereras då CTS=1 i USART_SR

Not: Kan inte användas med UART4 och UART5

Bit 9 CTSE: CTS enable

0: CTS handskakning deaktiverad

1: CTS handskakning aktiverad

Not: Kan inte användas med UART4 och UART5

Bit 8 RTSE: RTS enable

0: RTS handskakning deaktiverad

1: RTS handskakning aktiverad

Not: Kan inte användas med UART4 och UART5

Bit 7 DMAT: DMA enable transmitter

0: DMA mode deaktiveras för sändning.

1: DMA mode aktiveras för sändning.

Bit 6 DMAR: DMA enable receiver

0: DMA mode deaktiveras för mottagning

1: DMA mode aktiveras för mottagning

Bit 5 SCEN: Smartcard mode enable

0: Smartcard mode deaktiverad

1: Smartcard mode aktiverad

Bit 4 NACK: Smartcard NACK enable

Biten sätts och nollställs av programvara

0: NACK sändning vid paritetsfel deaktiverad

1: NACK sändning vid paritetsfel aktiverad

Bit 3 HDSSEL: Half-duplex selection

Aktivering av *single-wire half-duplex* mod

0: Half duplex mod är deaktiverad

1: Half duplex mod är aktiverad

Bit 2 IRLP: IrDA low-power

Biten väljer normal eller lågenergi *IrDA* mod

0: Normal mod

1: Lågenergi mod

Bit 1 IREN: IrDA mode enable

Biten aktiverar *IrDA* mod

0: IrDA deaktiverad

1: IrDA aktiverad

Bit 0 EIE: Error interrupt enable

Avbrottsmask för hela USART-kretsen.

0: Inga avbrott genereras

1: Avbrott genereras då DMAR=1 i USART_CR3 och FE=1 eller ORE=1 eller NF=1 i USART_SR.

C QUICK REFERENCE GUIDE

Strukturen hos ett C Program

```
/* Variabeldeklarationer */
(typ)(identifierare);
/* Definition av funktioner */
```

Definition av en funktion

```
(returvärde)(identifierare)(parameterlista)
{
(Definition av lokala variabler);
(programkod);
}
```

Motsvarande funktionsdeklaration:

```
extern
(returvärde)(identifierare)(parameterlista);
```

Variabeldefinition:

```
(typ) (identifierare {,identifierare} );
```

Variabeldeklaration:

```
(typ) (identifierare {,identifierare} );
```

Kommentarer

```
/*( kommentarer ) */
// Resten av raden är en kommentar
```

Typdefinition

```
typedef(befintlig typ)(nytt typnamn);
```

Exempel:

```
typedef int KILOGRAMS;
```

Funktionspekare:

```
typedef returtyp (fp*)(typ {,typ} );
```

Sammansatta typer:

Post

```
struct(tag)
{
    (type)(member name);
    ...
}(variable name);
typedef struct(tag)
{
    (type)(member name);
    ...
}(struct type name);
```

Union

```
union(tag)
{
    (type)(member name);
    ...
}(variable name);
typedef union(tag)
{
    (type)(member name);
    ...
}(union type name);
```

Operatörer

Företräde	Operator	Assoc.
15	() [] . ->	V
14	! ~ ++ -- + - * & (typ) sizeof	H
13	* / %	V
12	+ -	V
11	<< >>	V
10	< <= > >=	V
9	== !=	V
8	&	V
7	^	V
6		V
5	&&	V
4		V
3	?:	V
2	= *= /= %= += -= &=	H
1	^= = <<= >>=	V

Aritmetikoperatörer

Operator	Operation	Företräde
+	addition	12
-	subtraktion	12
*	multiplikation	13
/	division (heltalsresultat)	13
%	restdivision	13
+(unärt)	påverkar ej operand	14
-(unärt)	negera operand	14
++	inkrement, addera 1 till operand	14
--	dekrement, subtrahera 1 från operand	14

Relationsoperatörer

Operator	Operation	Företräde
==	likhet	9
!=	olikhet	9
>	större än	10
<	mindre än	10
>=	större än eller likhet	10
<=	mindre än eller likhet	10

Bitoperatörer

Operator	Operation	Företräde
&	bitvis OCH	9
	bitvis ELLER	9
^	bitvis EXKLUSIVT ELLER	10
~	bitvis komplement	10
<<	vänsterskift	10
>>	högerskift	10

Logikoperatörer

Operator	Operation	Företräde
&&	logiskt OCH	5
	logiskt ELLER	4
!	logisk negation	14

Programflödeskontroll

for-satsen används ofta för att skapa bundna programslingor.

```
for( uttryck1; uttryck2; uttryck3 )
    { satser }
```

Exempel:

```
for( i = 0; i < MAX; i++)
    { ... }
```

while-satsen skapar en iterativ slinga som upprepas så länge villkoret är sant.

```
while( uttryck )
{
    satser
}
```

do-while satsen är en variant, inte alls lika vanlig som *while*, men likafullt lämplig för vissa speciella programkonstruktioner.

```
do
{
    satser
} while( uttryck );
```

if-satsen utgör den enklaste flödeskontrollkonstruktionen:

```
if ( uttryck )
{
    satser
}
```

if-else satsen utvidgar med ett alternativt val, då villkoret är falskt.

```
if ( uttryck )
{
    satser
}else{
    satser
}
```

?-operator kan användas som ersättning för vissa *if/else*-satser.

Exempel:

```
c = (a<b)? a : b ;
```

är likvärdigt med:

```
if (a<b) c = a,
else c = b;
```

Med *switch-case* satsen konstruerar man enkelt flerval-konstruktioner:

```
switch( uttryck )
{
    case n:
    ...
    default:
    ...
}
```

Vanliga preprocessordirektiv

#define

I C används preprocessordirektivet **#define** för att definiera makron och symboliska konstanter. Makron är de identifierare som definieras av **#define** och som ersätts med deras värde före kompilering.

Exempel:

```
#define MACRO_NAME värde
#define MACRO_NAME ( uttryck )
Argument till makrot kan användas i uttrycket
#define MACRO_NAME(x, y,..) ( uttryck )
```

#include

Direktivet används för att inkludera innehållet i en specifik fil i källkoden före kompilering. Det låter dig använda funktioner, konstanter och variabler från externa bibliotek eller headerfiler. Det finns två typer:

```
#include <filnamn>
#include "filnamn"
```

Här anger filinkludering med dubbla citattecken (") att kompilatorn ska söka efter headerfilen i källfilens katalog medan <> används för systembibliotek.

#ifdef/#endif

Direktivet kontrollerar om ett makro är definierat. Om det är det inkluderas koden i **#ifdef**-blocket i programmet.

Exempel:

```
#define DEBUG
...
#ifdef DEBUG
    printf("We are debugging...");
#endif
```

#ifndef

Direktivet kontrollerar om ett makro är odefinierat. Om det *inte* är definierat inkluderas koden i **#ifndef**-blocket.

Exempel:

```
#ifndef DEBUG
    printf("Debugging is disabled...");
#endif
```

#if/#else/#elif

Dessa direktiv samverkar för att styra vilka delar av programmet som kompileras baserat på vissa villkor.

Om villkoret efter **#if** är sant, kompileras raderna efter det.

Om inte, kontrolleras villkoret efter tillhörande **#elif**. Om det är sant, kompileras dessa rader.

Om ingetdera villkoret är sant, kompileras raderna efter **#else**.

Exempel:

```
#define DEBUGLEVEL 2
...
#if DEBUGLEVEL > 3
    printf("Debuglevel=3");
#elif DEBUGLEVEL == 3
    printf("Debuglevel=2");
#else
    printf("Unknown Debuglevel");
#endif
```