

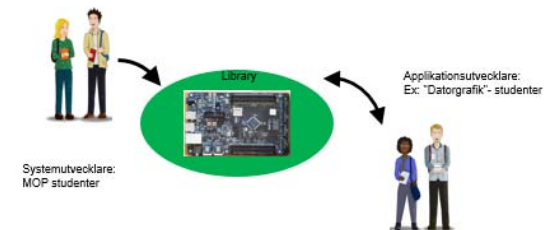
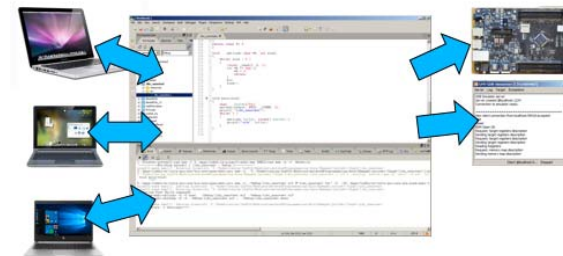
Programbibliotek

Ur innehållet:

Exekveringsmiljö

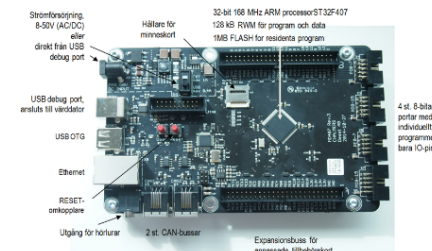
Olika typer av programbibliotek

- Kompilatorbibliotek
- C-bibliotek



Läsanvisningar:

Arbetsbok kapitel 8



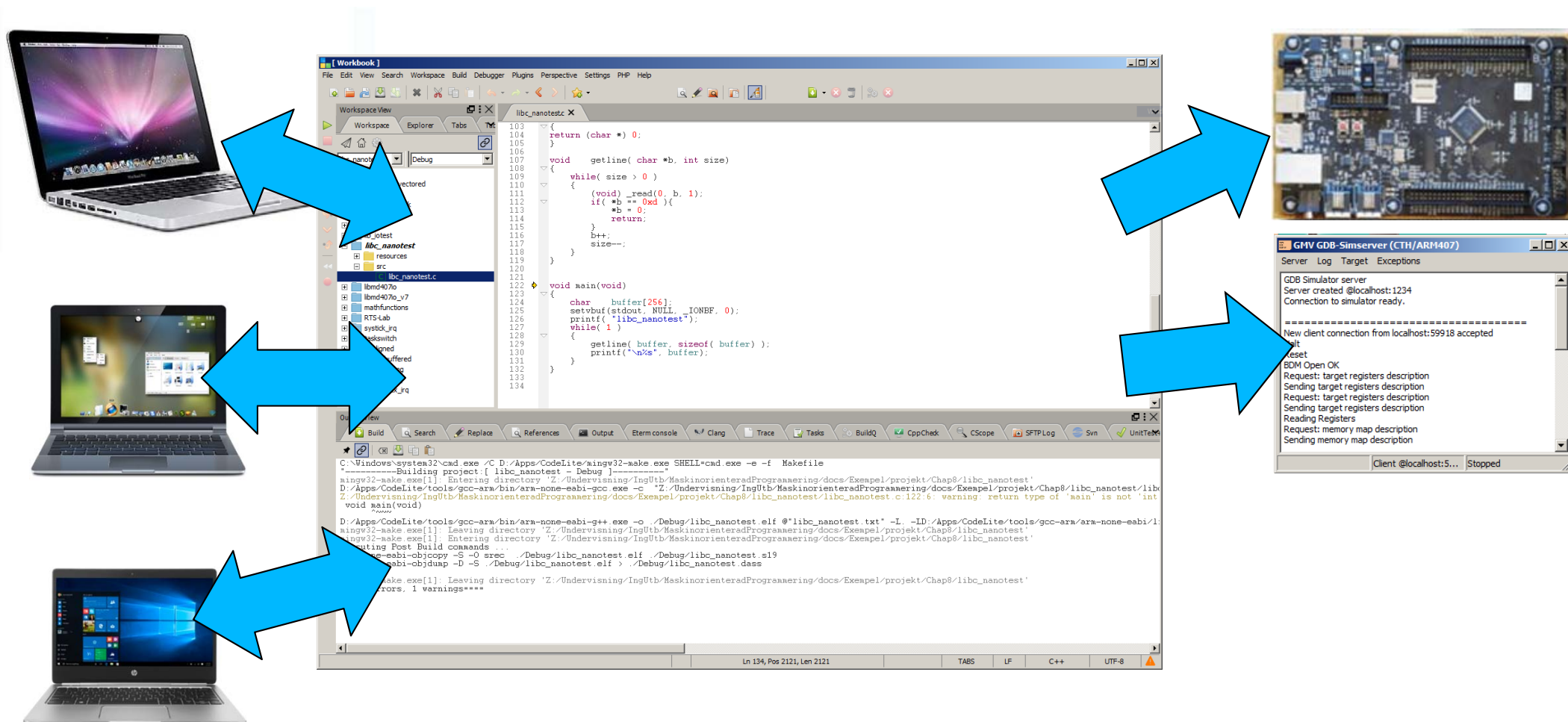
Målsättningar:

Att kunna använda färdiga programbibliotek

Att kunna skapa och underhålla ett nytt programbibliotek

Värddator och måldator:

Konventionell programutveckling sker på värddator *för värddator*.
Korsutveckling sker på värdator *för måldator*, i vårt fall MD407.

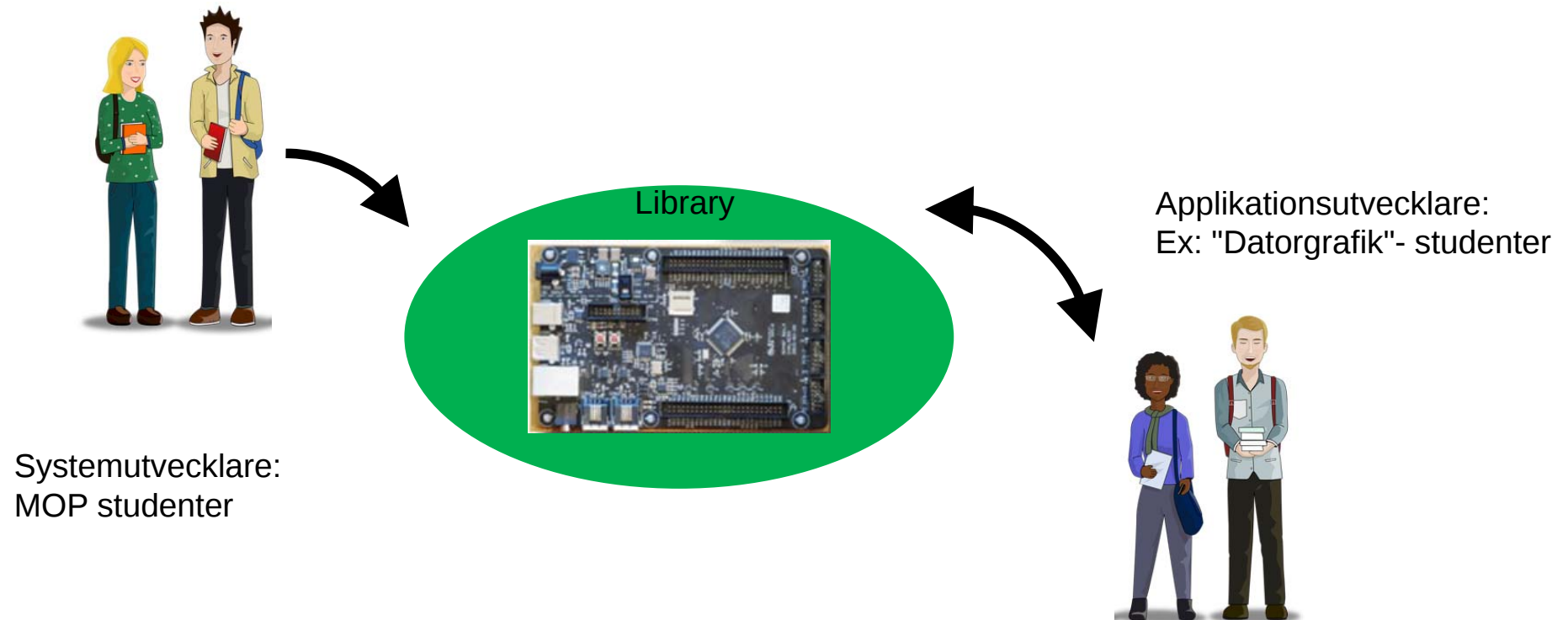


System-/Applikations- programmering för MD407

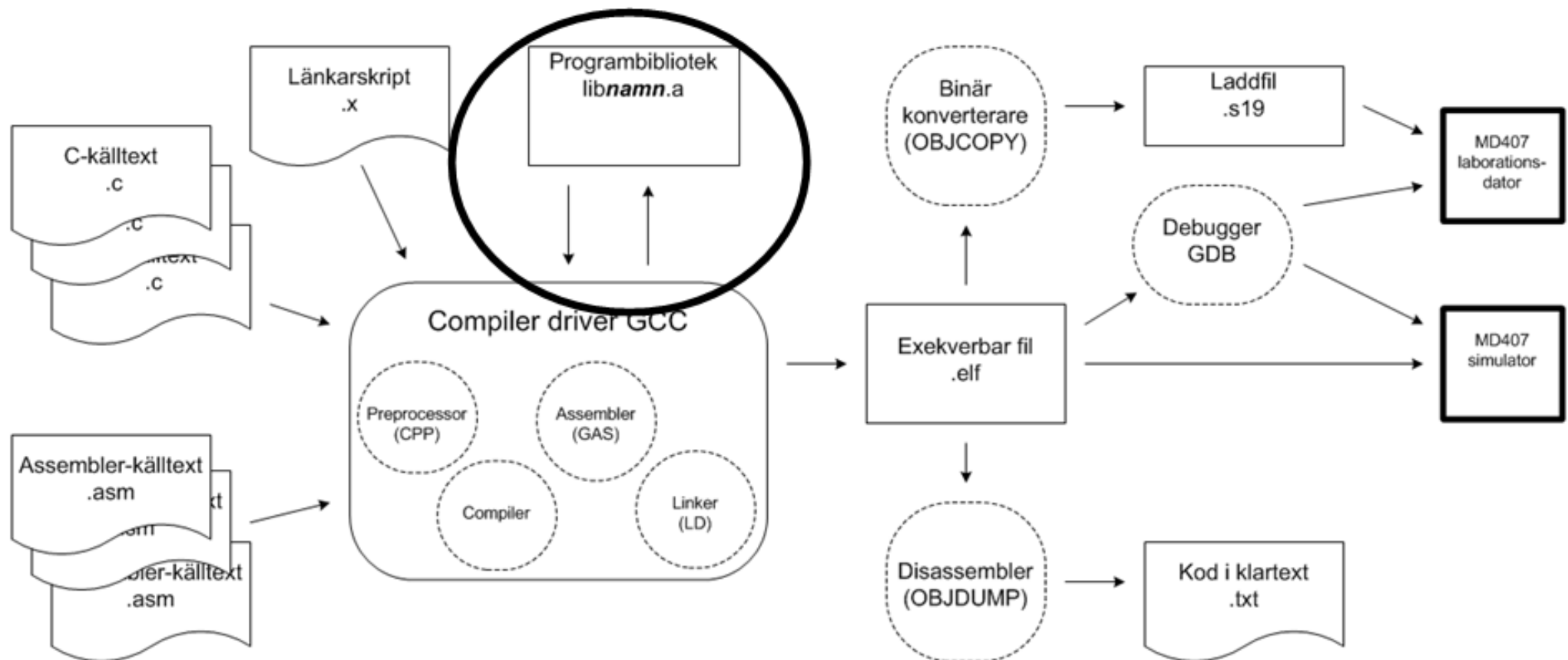
Med *programbibliotek*, avser vi att skapa grundläggande funktionalitet för användningen av vårt datorsystem. *Exempel: implementering av grundläggande operationer i programvara. In-/ut-matning, filhantering, fönsterhantering etc.*

Med *systemprogramvara* menar vi oftast "basal" programvara för vår dator, ett programlager med generellt användbar natur. *Exempel: operativsystem och drivrutiner.*

Med *applikationsprogramvara*, menar vi program avsedda för mer specifik användning. *Exempel: Spel, navigation, övervakning/styrning etc.*



Programbibliotek



Programbibliotek – olika typer

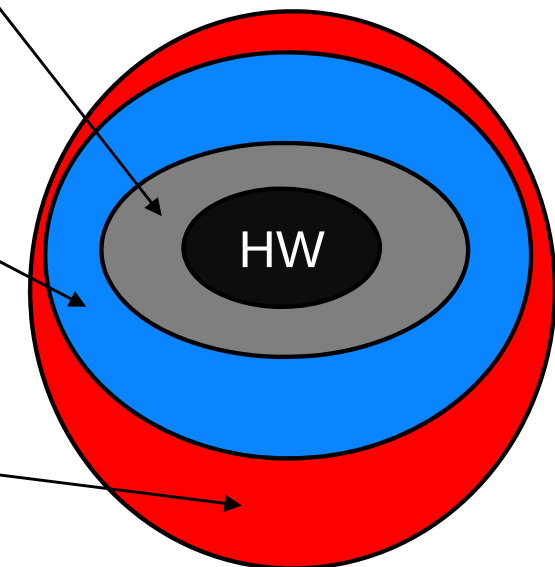
Programbibliotek kan ofta ses som "kompatibilitetslager" där en anpassning sker till någon speciell maskinvara och något specifikt operativsystem.

- Kompilatorbibliotek – tillhandahåller det stöd som krävs för att ett standardiserat C-program korrekt ska kunna översättas till maskinkod, oavsett processorarkitektur.
- **Standard C-bibliotek** – tillhandahåller en lång rad funktioner användbara i de flesta applikationer. Utgör också vanligtvis det viktigaste gränssnittet mot det använda operativsystemet.
- **Användarspecifika bibliotek.**

Exempel:
Flyttalsoperationer

Exempel:
printf...

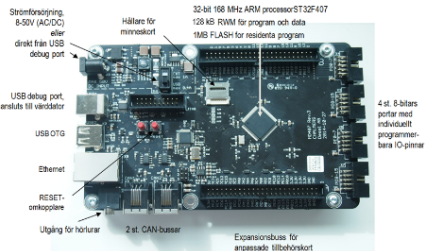
Exempel: GUI: Fönster och menyer





- Arkitektur – ARM-Cortex, tillhandahåller processorer som hanterar olika typer av instruktionsuppsättningar:

- | Instruktion | Storlek | Cortex M0 | Cortex M0+ | Cortex M1 | Cortex M3 | Cortex M4 | Cortex M7 | Ark. |
|--|---------|-----------|------------|-----------|-----------|-----------|-----------|-----------|
| ADC, ADD, (ADR), AND, ASR, B, BIC, BKPT, BLX, BX, CMN, CMP, CPS, EOR, LDM, (LDMIA, LDMFD), LDR, LDRB, LDRH, LDRSB, LDRSH, LSL, LSR, MOV, MUL, MVN, NOP, ORR, POP, PUSH, REV, REV16, REVSH, ROR, RSB, SBC, SEV, STM, (STMTA, STMEA), STR, STRB, STRH, SUB, SVC, SXTB, SXTH, TST, UXTB, UXTH, WFE, WFI, YIELD | 16-bit | x | x | x | x | x | x | v6 |
| BL, DMB, DSB, ISB, MRS, MSR | 32-bit | x | x | x | x | x | x | |
| CBNZ, CBZ, IT | 16-bit | | | | x | x | x | |
| ADC, ADD, AND, ASR, B, BFC, BFI, BIC, CDP, CLREX, CLZ, CMN, CMP, DBG, EOR, LDC, LDMA, LDMBD, LDR, LDRB, LDRBT, LDRD, LDREX, LDREXB, LDREXH, LDRH, LDRHT, LDRSB, LDRSBT, LDRSH, LDRT, MCR, LSL, LSR, MLS, MCRR, MLA, MOV, MOVT, MRC, MRR, MVL, MVN, NOP, ORN, ORR, PLD, PLDW, PLI, POP, PUSH, RBIT, REV, REV16, REVSH, ROR, RRX, RSB, SBC, SBFX, SDIV, SEV, SMLAL, SMULL, SSAT, STC, STMBD, STR, STRB, STRBT, STRD, STREX, STREXH, STREXH, STRH, STRHT, STRT, SUB, SXTB, SXTH, TBB, TBH, TEQ, TST, UBFX, UDIV, UMLAL, UMULL, USAT, UXTB, UXTH, WFE, WFI, YIELD | 32-bit | | | | x | | x | v7 |
| PKH, QADD, QADD16, QADD8, QASX, QDADD, QDQSUB, QDSAX, QSUB, QSUB16, QSUB8, SADD16, SADD8, SASX, SEL, SHADD16, SHADD8, SHASX, SHSAX, SHSUB16, SHSUB8, SMLABB, SMLABT, SMLATB, SMLATT, SMLAD, SMLALBB, SMLALBT, SMLALTB, SMLALTT, SMLALD, SMLAWB, SMLAWT, SMLSLD, SMLSLD, SMLMA, SMLMS, SMMUL, SMUAD, SMULBB, SMULBT, SMULTT, SMLT, SMULWT, SMULWB, SMUSD, SSAT16, SSAX, SSUB16, SSUB8, XTAB, XTAB16, XTAHB, SXTAB, SXTB16, UADD16, UADD8, UASX, UHADD16, UHADD8, UHASX, UHSAX, UHSUB16, UHSUB8, UMAL, UQADD16, UQADD8, UQASX, UQASX, UQSUB16, UQSUB8, USAD8, USADA8, USAT16, USAX, UCHB16, UHSUB, UXTAB, UXTAB16, UXTAH, UXTB16 | 32-bit | | | | | x | x | v7e (DSP) |
| VABS, VADD, VCMP, VCMPPE, VCVT, VCVTR, VDIV, VLDM, VLDR, VMLA, VMLS, VMOV, VMRS, VMSR, VMUL, VMUL, VMULA, VMULS, VMULL, VPOB, VPUSH, VSCB, VSTH, VSTR, VSUB | 32-bit | | | | | SP EDU | SP EDU | 32-b EDU |
| FP- dubbel precision | 32-bit | | | | | | DP | 64-b FP |



Måldator – minnesdisposition

Block 1 SRAM

- Minneskonfiguration – Hur mycket minne finns det och var är det placerat i adressrummet?

Adressrum	
Tillverkar-specifikt	FFFF FFFF
	E010 0000
	E00F FFFF
Periferbuss	
	E000 0000
	DEFF FFFF
Externa enheter	
	A000 0000
	9FFF FFFF
Extern RAM-minne	
	6000 0000
	5FFF FFFF
Periferikretsar	
	4000 0000
	3FFF FFFF
Block 1 SRAM	
	2000 0000
	1FFF FFFF
Block 0 kod	
	0000 0000

2001 FFFF

2001 C400

2001 C3FF

2001 C000

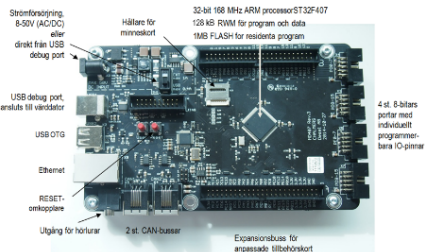
2001 BFFF

Monitor/
debugger
stack och data
Relokerade
avbrottsvektorer

Reserverat för
applikationen

Informationen måste finnas för att vi exempelvis ska kunna implementera dynamisk minneshantering med "malloc/free"

2000 0000



Måldator – periferienheter

- Tillgängliga periferienheter –
Systemenheter
IO-enheter

Adressrum

Tillverkar-specifikt	FFFF FFFF	
	E010 0000	
	E00F FFFF	
Periferibuss		
	E000 0000	
	DFFF FFFF	
Externa enheter		
	A000 0000	
	9FFF FFFF	
Externt RAM-minne		
	6000 0000	
	5FFF FFFF	
Perifrikretsar		
	4000 0000	
	3FFF FFFF	
Block 1 SRAM		
	2000 0000	
	1FFF FFFF	
Block 0 kod		
	0000 0000	

Periferibuss

E000 EF44		"Floating point unit"
E000 EF30		
E000 EF03		"NVIC"
E000 EF00		
E000 EDB8		"Memory protection unit"
E000 ED90		
E000 ED8B		"Floating point unit, access control"
E000 ED88		
E000 ED3F		"System Control Block"
E000 ED00		
E000 E4EF		"NVIC"
E000 E100		
E000 E01F		"System timer"
E000 E010		

Perifrikretsar

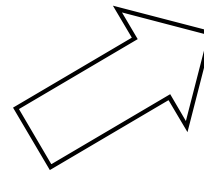
A000 0000 - A000 0FFF	FSMC control reg.	AHB3
5006 0800 - 5006 0BFF	RNG	
5006 0400 - 5006 07FF	HASH	
5006 0000 - 5006 03FF	CRYP	
5005 0000 - 5005 03FF	DCMI	
5000 0000 - 5003 FFFF	USB OTG FS	
4004 0000 - 4007 FFFF	USB OTG HS	
4002 8000 - 4002 8BFF	DMA2D	
4002 9000 - 4002 93FF		
4002 8C00 - 4002 8FFF		
4002 8800 - 4002 8BFF	ETHERNET MAC	
4002 8400 - 4002 87FF		
4002 8000 - 4002 83FF		
4002 6400 - 4002 67FF	DMA1	
4002 6000 - 4002 63FF	DMA2	
4002 4000 - 4002 4FFF	BKPSRAM	
4002 3C00 - 4002 3FFF	Flash interface	
4002 3800 - 4002 3BFF	RCC	
4002 3000 - 4002 33FF	CRC	
4002 2800 - 4002 2BFF	GPIOK	
4002 2400 - 4002 27FF	GPIOJ	
4002 2000 - 4002 23FF	GPIOI	
4002 1C00 - 4002 1FFF	GPIOH	
4002 1800 - 4002 1BFF	GPIOG	
4002 1400 - 4002 17FF	GPIOE	
4002 1000 - 4002 13FF	GPIOF	
4002 0C00 - 4002 0FFF	GPIOC	
4002 0800 - 4002 0BFF	GPIOB	
4002 0400 - 4002 07FF	GPIOA	
4002 0000 - 4002 03FF	LCD-TFT	
4001 6800 - 4001 6BFF	SAI1	
4001 5800 - 4001 5BFF	SPD1	
4001 5400 - 4001 57FF	SPD2	
4001 5000 - 4001 53FF	SPD3	
4001 4800 - 4001 4BFF	TIM11	
4001 4400 - 4001 47FF	TIM12	
4001 4000 - 4001 43FF	TIM9	
4001 3C00 - 4001 3FFF	EXTI	
4001 3800 - 4001 3BFF	SYSCFG	
4001 3400 - 4001 37FF	SPI4	
4001 3000 - 4001 33FF	SPI1	
4001 2C00 - 4001 2FFF	SDIO	
4001 2000 - 4001 23FF	ADC1-ADC3-ADC3	
4001 1400 - 4001 17FF	USART6	
4001 1000 - 4001 13FF	USART1	
4001 0400 - 4001 07FF	TIM8	
4001 0000 - 4001 03FF	TIM1	
4000 7C00 - 4000 7FFF	UART8	
4000 7800 - 4000 7BFF	UART7	
4000 7400 - 4000 77FF	DAC	
4000 7000 - 4000 73FF	PWR	
4000 6800 - 4000 6BFF	CAN2	
4000 6400 - 4000 67FF	CAN1	
4000 5C00 - 4000 5FFF	IC2	
4000 5800 - 4000 5BFF	IC2	
4000 5400 - 4000 57FF	IC1	
4000 5000 - 4000 53FF	UART5	
4000 4C00 - 4000 4FFF	UART4	
4000 4800 - 4000 4BFF	USART3	
4000 4400 - 4000 47FF	USART2	
4000 4000 - 4000 43FF	IC3ext	
4000 3C00 - 4000 3FFF	SPI3/S2S3	
4000 3800 - 4000 3BFF	SPI2/S2S2	
4000 3400 - 4000 37FF	IC2ext	
4000 3000 - 4000 33FF	WDG	
4000 2C00 - 4000 2FFF	WWDG	
4000 2800 - 4000 2BFF	RTC & BKP Reg	
4000 2000 - 4000 23FF	TIM14	
4000 1C00 - 4000 1FFF	TIM13	
4000 1800 - 4000 1BFF	TIM12	
4000 1400 - 4000 17FF	TIM7	
4000 1000 - 4000 13FF	TIM6	
4000 0C00 - 4000 0FFF	TIM5	
4000 0800 - 4000 0BFF	TIM4	
4000 0400 - 4000 07FF	TIM3	
4000 0000 - 4000 03FF	TIM2	

Informationen måste finnas för att vi exempelvis ska kunna implementera "drivrutiner" för enheterna

Kompilatorbibliotek

gencode.c

```
int    a,b,c;
void   f( void )
{
    a = b/c;
}
```



gencodev6.dass (armv6-m)

```
ldr    r0,b
ldr    r1,c
bl     __divsi3
ldr    r3,=a
str    r0,[r3]
```

gencodeV7.dass (armv7-m)

```
ldr    r2,b
ldr    r3,c
sdiv   r3,r2,r3
ldr    r2,=a
str    r3,[r2]
```

lib/gcc/arm-none-eabi/.../...libgcc.a

```
1000101001110001100101
__divsi3:
001011000010100010010010010
10001010011100011.....
```

Funktionerna i kompilatorbiblioteket används bara internt för kodgenerering, dvs. är ej publika och det behövs därför ingen header-fil med deklARATIONER.
Information om processorarkitektur måste vara konsistent vid kompilering och länkning.

Kompilatorbibliotek

I Grundläggande datorteknik har vi studerat algoritmer för multiplikation och division. Dessa kan användas för mjukvaruimpementering av operationerna.

EXEMPEL 4.39 ROBERTSONS METOD FÖR MULTIPLIKATION AV TAL MED TECKEN, $X < 0, Y < 0$.

Algorithm: Multiplikation $P(\text{produkt}) = X(\text{multiplikand}) \times Y(\text{multiplikator})$
 Partialprodukt 0, $PP(0) = 0$
 med början vid multiplikatorns LSB (y_0)
 för varje bit i hos multiplikatorn
 Om $y_i = 1$, addera multiplikand till nästa partialprodukt
 annars addera 0 till nästa partialprodukt
 skifta resultatet ett steg till höger

PP(3)	11101	110	$\times 2$ (skifta aritmetiskt)
	+00110		$y_3=1 \Rightarrow \text{ADD } -X$
	00011	110	
P= 30 =PP(4)=P	0001	1110	

EXEMPEL 4.42 UTFÖR BINÄR DIVISION AV 13/5 MED ÅTERSTÄLLNINGSMETOD.

Svaret ska anges med 4 kvotbitar och 4 restbitar. Varje steg i algoritmen ska redovisas.

ikna
 } bitar ska skiftas):

R(2) $\times 2 =$	0110100
$+ (-Y)$	1011 steg 3: pröva

Algorithm: Division med återställning

$$R = X - Q \times Y$$

$$Q = q_0 q_1 q_2 \dots q_{n-1}$$

n =antal kvotbitar att beräkna, partialrester $R(i)$ bestäms enligt

$$R(0) = X$$

$$R(1) = R(0) - q_0 \times Y \quad q_0 = 1 \text{ om } R(1) \geq 0, q_0 = 0 \text{ annars}$$

för $i = 2..n$

$$R(i) = 2 \times R(i-1) - q_i \times Y \quad q_i = 1 \text{ om } R(i) \geq 0, q_i = 0 \text{ annars}$$

I algoritmerna används enklare operationer: skift, addition och subtraktion.

Kompilatorbibliotek

Representation av flyttal

Ett flyttal uttrycks allmänt som:

$$(-1)^S M \times 2^E$$

där:

S (*sign*) är teckenbiten för flyttalet

$S=0$ anger ett positivt flyttal ty $(-1)^0 = 1$

$S=1$ anger ett negativt flyttal ty $(-1)^1 = -1$

M utgör talets mantissa

E utgör talets exponent.

Exponenten väljs från någon representation med inbyggt tecken. Det är värt att notera att för ett normaliserat flyttal på binär form gäller att:

$$(M)_2 = 1.xxxxx$$

dvs. mantissans första siffra är alltid 1. Detta innebär att vi, då vi lagrar flyttal, kan utelämna denna siffra och på så vis åstadkomma ett kompaktare format.

Exempel: Omvandling av heltal till IEEE flyttal

Vi vill skriva talet $(2,52)_{10} \cdot 10^4$ som ett IEEE754-single format flyttal:

Vi har tidigare kommit fram till resultatet:

$$(2,52)_{10} \cdot 10^4 = (1.100\ 0100\ 1110\ 000)_2 \times 2^{14}$$

Mantissan ska ha totalt 24 bitar, vi "fyller på" med nollor på slutet...

$$M = (1.100\ 0100\ 1110\ 0000\ 0000\ 0000)_2$$

Signifikanden F dvs. den del av mantissan som ska lagras får stryka den mest signifikanta ettan, vi har då:

$$F = (100\ 0100\ 1110\ 0000\ 0000\ 0000)_2$$

Exponenten i IEEE-formen uttrycks av karakteristikan E' , excess(127) kod, dvs. $E' = E + 127$, där E betecknar exponenten i talet vi utgår från (2^{14}) dvs. $E=14$ varför

$$E' = 14 + 127 = 141 = (1000\ 1101)_2$$

eftersom talet är positivt får vi $S = 0$. Vi sammanställer nu resultatet i 32-bitars form (*SFP*) och får slutligen:

$$(2,52)_{10} \cdot 10^4 = (0100\ 0110\ 1100\ 0100\ 1110\ 0000\ 0000\ 0000)_{SFP}$$

Representationen av heltal och flyttal är olika men de lagras med samma storlek.

```
float f;    int i; /* båda typerna är 32 bitar */
```

Det innebär exempelvis att tilldelningen:

```
f = (float) i;
```

ska göras enligt exemplet ovan...

Kompilatorbibliotek

Flaggor till kompilatorn:

`-mthumb; -march=armv6-m; -msoft-float`

ger koden:

```
ldr    r0, i
bl    __aeabi_i2f
ldr    r1, =f
str    r0, [r1]
```

```
float f;
int i;

...
f = (float) i;
```

Flaggor till kompilatorn:

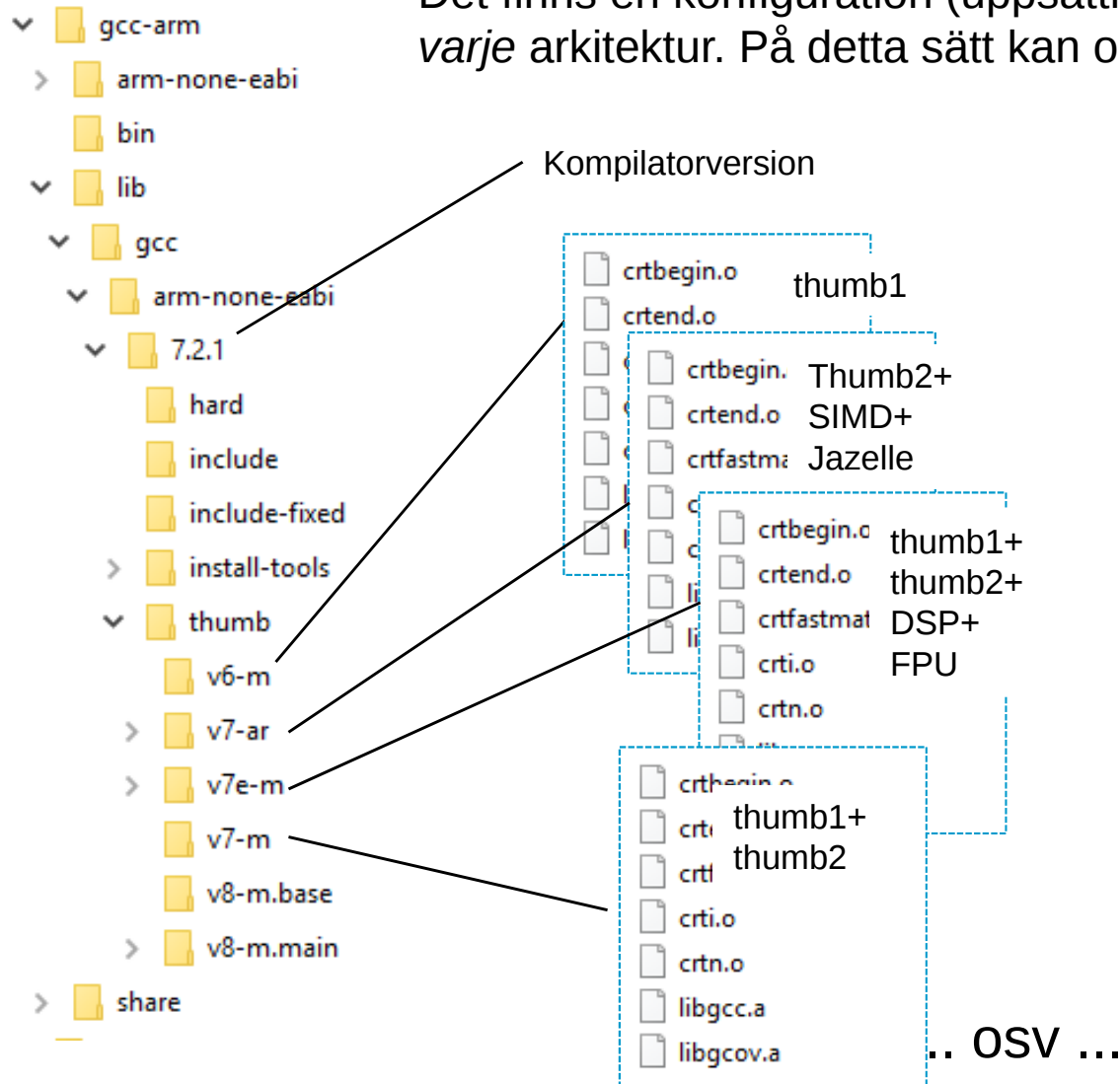
`-mfloat-abi=hard; mthumb; -mfpu=fpv4-sp-d16; -march=armv7-m;`

ger koden:

```
ldr                r0, i
vmov               s15, r0
vcvt.f32.s32     s15, s15
ldr                r0, =f
vstr.32            s15, [r0]
```

Kompilatorbibliotek - olika konfigurationer

Det finns en konfiguration (uppsättning kompilator-bibliotek och startfiler) för *varje* arkitektur. På detta sätt kan optimal kod länkas till applikationen.



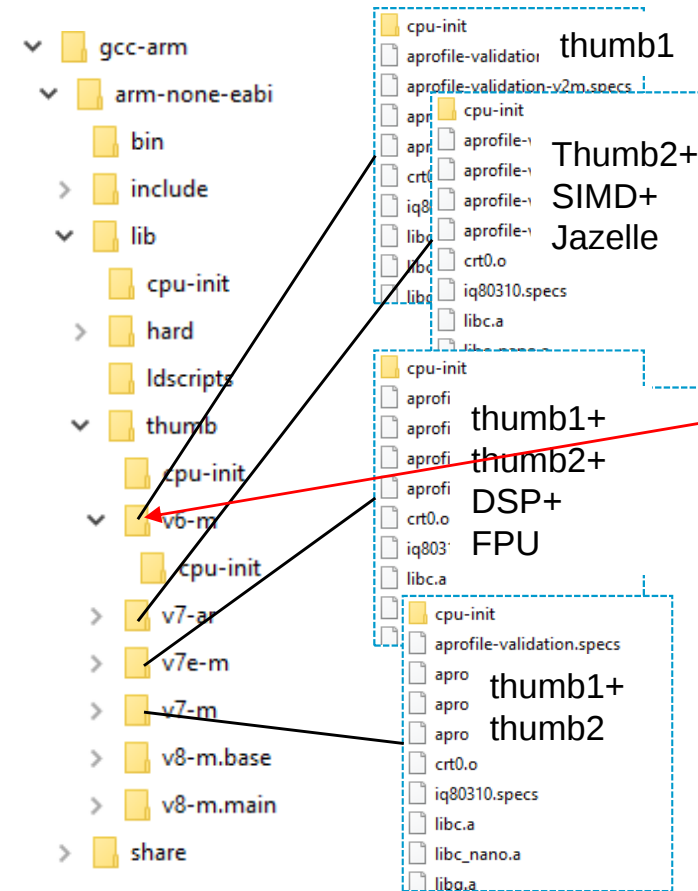
RTE - Run Time Environment
(exekveringsmiljö)

crtbegin, crtend etc..
(CRT= C Run Time)
Kallas också "startfiles"

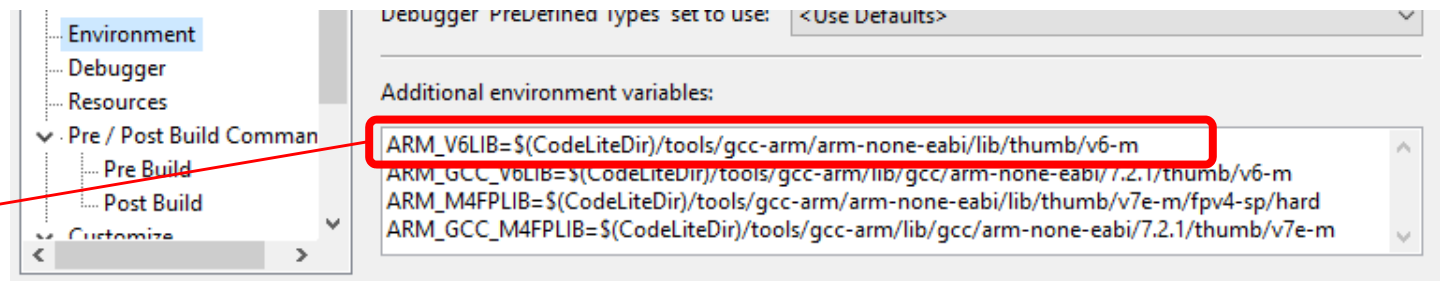
Libgcc, kompatibilitetskod

För att *undvika* länkning med
standard filer ger man länkarflaggan
-nostartfiles

C-bibliotek – olika konfigurationer



På samma sätt finns det en konfiguration (uppsättning C-bibliotek) för *varje* arkitektur. Vi använder miljövariabler för att precisera biblioteken:

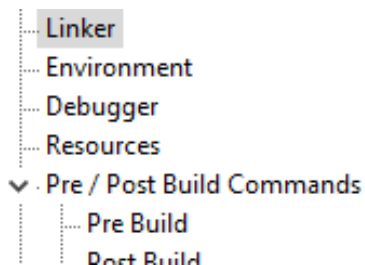


I kommandoraden för länkaren kan vi sedan enkelt välja konfiguration. I listan av bibliotek utelämnar man (konvention) delar av namnet, dvs:

`libgcc.a` skrivs `gcc`

`libc_nano.a` skrivs `c_nano`

Vi väljer också att använda vår egen "run-time-miljö" (**-nostartfiles**)



Options	
Use with global settings	append
Linker Options	-T\$(ProjectPath)/md407-ram.x;-L\$(ARM_V6LIB) -L\$(ARM_GCC_V6LIB);-nostartfiles;
Libraries Search Path	
Libraries	gcc;c_nano

Standard C biblioteket

"The C standard library" eller libc utgör standardbiblioteket för programspråket C och specificerades ursprungligen i ANSI C standarden från 1989, (C89). Flera tillägg har gjorts efter detta.

<code><assert.h></code>	Definierar macro som kan användas för att upptäcka logiska fel och andra felaktigheter i "debug"-versioner av program.
<code><ctype.h></code>	Deklarerar funktioner för att klassificera och omvandla tecken.
<code><errno.h></code>	För att kontrollera felkoder från biblioteksfunktionerna.
<code><float.h></code>	Definierar olika konstanter som definierar implementationsspecifika detaljer i flyttalshanteringen.
<code><limits.h></code>	Definierar olika konstanter som definierar implementationsspecifika detaljer i heltalshanteringen.
<code><locale.h></code>	Deklarerar funktioner för lokala anpassningar.
<code><math.h></code>	Deklarerar vanliga matematiska funktioner..
<code><setjmp.h></code>	Deklaration för speciella programflödesändringar.
<code><signal.h></code>	Deklarerar funktioner för hantering av "signaler".
<code><stdarg.h></code>	För variabelt antal argument till en funktion.
<code><stddef.h></code>	Definierar diverse olika typer och macron.
<code><stdio.h></code>	Deklarerar in- och utmatningsfunktioner.
<code><stdlib.h></code>	Deklarerar funktioner för numeriska konverteringar, slumpvalsgenerering och minneshantering.
<code><string.h></code>	Deklarerar funktioner för stränghantering.
<code><time.h></code>	Deklarerar funktioner för datum och tid.

Dynamisk minneshantering, malloc/free

C-biblioteket tillhandahåller rutiner som `malloc` och `free` för dynamisk minneshantering men har ingen information om hur måldatorns minne disponeras. Runtime biblioteket måste därför definiera:

```
void *_sbrk (int increment)
```

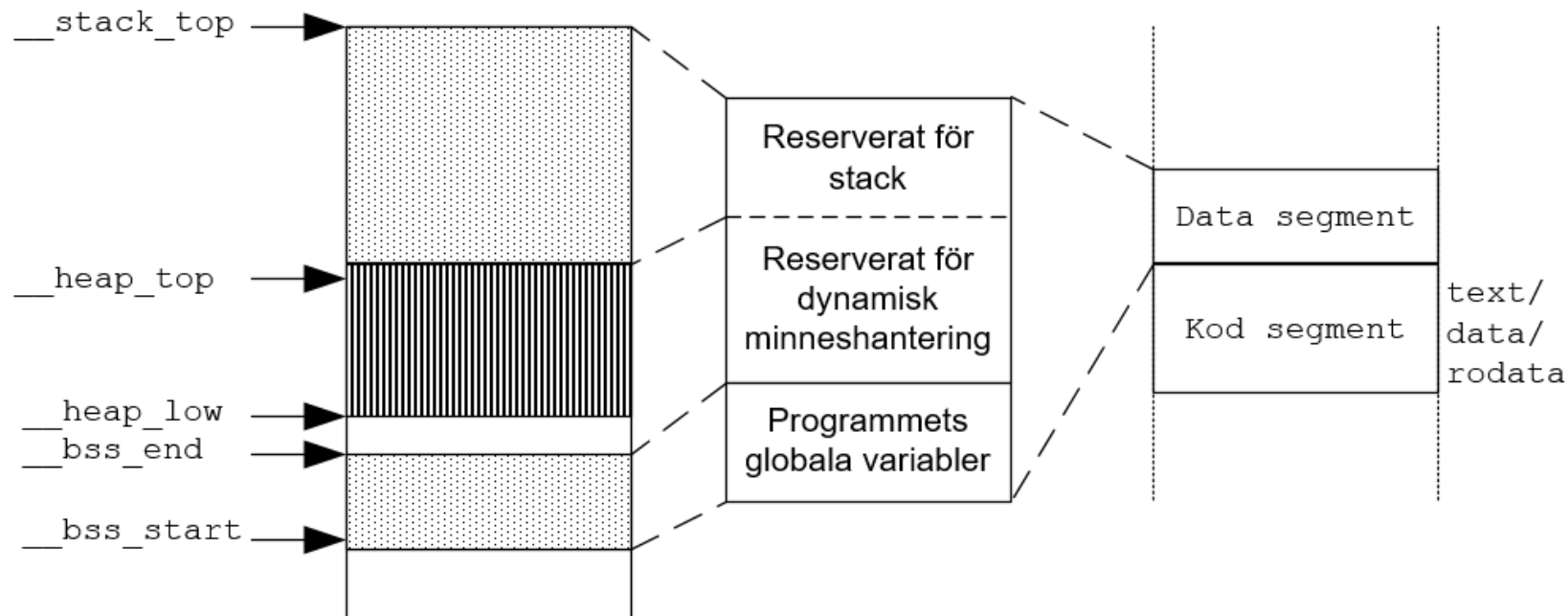
(*set program break*) som tillhandahåller adresser till minne som är tillgängligt för `malloc`.

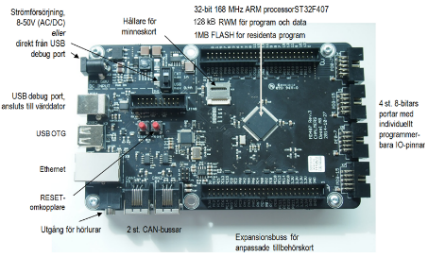
Länkaren skapar de symboler vi behöver för att administrera minnet

```
...
*(.start_section)

*(.text)

*(.data)
*(.rodata)
. = ALIGN(4);
__bss_start = .;
*(.bss)
__bss_end = .;
. = ALIGN(4096);
__heap_low = .;
. = . + 0x400;
__heap_top = .;
. = . + 0x400;
__stack_top = .;
```





Implementering av modifierad startup

Vår startup-sekvens för *md407* behöver modifieras som förberedelse för att använda C-biblioteket.

För applikationsprogram krävs att C-biblioteket initierats av någon funktion som utförts före **main**.

Standardprocedurer för *pre main* och *after main* finns crt (*c-run time*).

Detta motsvaras av den "startup" vi själva tidigare skapat i våra program.

```
void startup ( void )
{
    __asm volatile(" LDR    R0,=__stack_top\n");
    __asm volatile(" MOV    SP,R0\n");
    __asm volatile(" BL     crt_init\n");
    __asm volatile(" BL     main\n");
    __asm volatile(" BL     crt_deinit\n");
    __asm volatile("exit: B exit\n");
}
```

```
...
*(.start_section)
*(.text)
*(.data)
*(.rodata)
. = ALIGN(4);
__bss_start__ = .;
*(.bss)
__bss_end__ = .;
. = ALIGN(4096);
__heap_low__ = .;
. = . + 0x400;
__heap_top__ = .;
. = . + 0x400;
__stack_top = .;
```

Initiering av runtime funktioner

Standarden säger att icke initierade variabler (dvs. bss-arean) ska initieras till 0 av run-time systemet...

```
void crt_init() {  
    extern char __bss_start;  
    extern char __bss_end;  
    extern char __heap_low;  
    extern char __heap_top;  
    char *s;  
    s = &__bss_start;  
    while( s < &__bss_end)  
        *s++ = 0;  
    s = &__heap_low;  
    while( s < &__heap_top )  
        *s++ = 0;  
}
```

```
...  
    *(.start_section)  
    *(.text)  
    *(.data)  
    *(.rodata)  
    . = ALIGN(4);  
    __bss_start = .;  
    *(.bss)  
    __bss_end = .;  
    . = ALIGN(4096);  
    __heap_low = .;  
    . = . + 0x400;  
    __heap_top = .;  
    . = . + 0x400;  
    __stack_top = .;
```

Implementering av _sbrk

```
#include <errno.h>
static char *heap_end;
char * _sbrk( int incr) {
    extern char __heap_low;
    extern char __heap_top;
    char *prev_heap_end;
    if (heap_end == 0) {
        heap_end = &__heap_low;
    }
    prev_heap_end = heap_end;

    if (heap_end + incr > &__heap_top) {
        /* Heap and stack collision */
        errno = ENOMEM;
        return (char *)-1;
    }
    heap_end += incr;
    return (char *) prev_heap_end;
}
```

```
...
*(.start_section)

*(.text)

*(.data)
*(.rodata)
. = ALIGN(4);
__bss_start = .;
*(.bss)
__bss_end = .;
. = ALIGN(4096);
__heap_low = .;
. = . + 0x400;
__heap_top = .;
. = . + 0x400;
__stack_top = .;
```

Användning av standard C-biblioteket

TypdeklARATIONER för funktionerna i C-biblioteket har organiserats i olika header-filer.

main.c

```
#include <stdio.h> /* printf */
#include <stdlib.h> /* malloc */

void foo(){
    char *cp;
    cp = (char *) malloc( 100 );
    if( cp == (char *) 0 )
    {
        printf( "Can't malloc!");
        exit(-1);
    }
    free( cp );
}
```

arm-none-eabi/include/stdio.h

```
#ifndef _STDIO_H_
#define _STDIO_H_
...
int printf(const char *,...);
...
```

arm-none-eabi/lib/.../libc.a

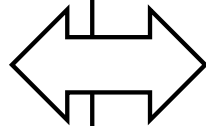
```
malloc:
1000101001110001100101 ....
printf:
001011000010100010010010010
10001010011100011.....
free: 10100111000110 ....
```

"C-nano" - lättviktsimplementering

Applikation...

```
#include <stdio.h>

void main(void)
{
    printf( "Hello World!");
}
```

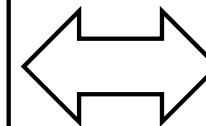


Standard C-bibliotek

libc_nano.a

```
100010100111000
1100101
printf:
001011000010100
010010010010100
01010011100011.
```

....



C-RunTime funktioner

?

Då vi länkar vår applikation upptäcker vi att en rad symboler saknas. Detta beror på att IO-funktioner i standard C-biblioteket också länkas mot *maskinberoende* funktioner (C-RunTime), som:

`_open`, `_close`, `_lseek`, `_read`, `_write`, `_fstat`, `_isatty`

För att kunna använda IO-funktionerna måste vi då först implementera dessa maskinberoende funktioner.

stdio – keypad och ASCII-display

Datastruktur: "Device driver".

```
#include <sys/stat.h>
```

```
typedef struct
{
    char    name[16];          /* Device name */
    int (*init) (int);
    void (*deinit) (int);
    int (*fstat)(struct stat *st);
    int (*isatty)(void);
    int (*open)(const char name,int flags,int mode);
    int (*close)(void);
    int (*lseek)(int ptr, int dir);
    int (*write)(char *ptr, int len);
    int (*read)(char *ptr, int len);
} DEV_DRIVER_DESC, *PDEV_DRIVER_DESC;
```

```
static DEV_DRIVER_DESC KeyPad =
{
    {"Keypad"},
    keypad_init,
    keypad_deinit,
    0,
    0,
    0,
    0,
    0,
    0,
    keypad_read
};
```

stdio – keypad och ASCII-display

"Device driver".

```
#define MAX_FILENO      5
#define KEYPAD_FILENO   3
#define ASCII_DISPLAY_FILENO  4

PDEV_DRIVER_DESC open_file_table[MAX_FILENO] =
{
    &StdIn,
    &StdOut,
    &StdErr,
    &KeyPad,
    &AsciiDisplay,
};

void crt_init() {
    ...
    for( int i = 0; i < MAX_FILENO; i++ )
    {
        fd = open_file_table[i];
        if( fd->init != 0 )
            (void) fd->init( 0 );
    }
}
```

```
static DEV_DRIVER_DESC KeyPad =
{
    {"Keypad"},
    keypad_init,
    keypad_deinit,
    0,
    0,
    0,
    0,
    0,
    0,
    keypad_read
};
```

"C-run time" - lättviktsimplementering för MD407

_fstat, returnera information om en öppen fil.

```
#include <sys/stat.h>
int _fstat(int file, struct stat *st) {
    st->st_mode = S_IFCHR;
    return 0;
}
```

`struct stat` och
`S_IFCHR` deklareras i
`sys/stat.h`

S_IFCHR, anger att detta är en tecken orienterad fil, andra exempel är:

`S_IFDIR`, filen är ett bibliotek.

`S_IFBLK`, filen är block-orienterad, (typiskt på disk) osv...

Vår implementering kommer endast att stödja de speciella filerna `stdin`, `stdout` och `stderr`, som teckenorienterade filer, dvs. in- och utmatning via en terminal.

```
int _isatty(int file) { return 1; }
```

`stdin`, `stdout` och `stderr`, kan normal sett dirigeras om till blockorienterade enheter, dock inte i vår implementering...

"C-run time" - lättviktsimplementering för MD407

_open, öppna en fil för läsning och/eller skrivning.

I vår implementering gör vi det enkelt och låter systemet öppna filer för enheterna, det räcker då att vår implementering returnerar en felkod.

```
int _open(const char *name, int flags, int mode) { return -1; }
```

_close, stäng en fil som är öppen för läsning och/eller skrivning.

Ej tillämbart, våra öppna enheter stängs av operativsystemet.

```
int _close(int file) { return -1; }
```

_lseek, sök till position i en fil som är öppen för läsning och/eller skrivning.

Måste vara en blockorienterad fil, ej tillämbart för teckenorienterade enheter.

```
int _lseek(int file, int ptr, int dir) { return 0; }
```

"C-run time" - lättviktsimplementering för MD407

_write, skriv till en öppen fil.

Vår implementering är enklast tänkbara . Man hade kunnat lägga in felkontroll här eftersom file ska vara någon av stdout eller stderr....

```
int _write(int file, char *ptr, int len) {  
    PDEV_DRIVER_DESC drvr;  
    drvr = open_file_table[file];  
    if(drvr == 0)  
        return 0;  
    return drvr->write(ptr, len);  
}
```

_read, läs från en öppen fil.

```
int _read(int file, char *ptr, int len) {  
    PDEV_DRIVER_DESC drvr;  
    drvr = open_file_table[file];  
    if(drvr == 0)  
        return 0;  
    return drvr->read(ptr, len);  
}
```

Skapa run-time bibliotek för MD407

EXEMPEL

För att slippa kompilera vårt lilla runtime-bibliotek *tillsammans* med applikationen skapar vi i stället ett programbibliotek `libMD407.a`

```
/*
 * libMD407.c
 * MD407 library
 */

/* declarations goes to 'libMD407.h' */
#include "libMD407.h"

/* Define variables here */
static char *heap_end;

/* Define functions here */
void crt_init() {
...
}
```

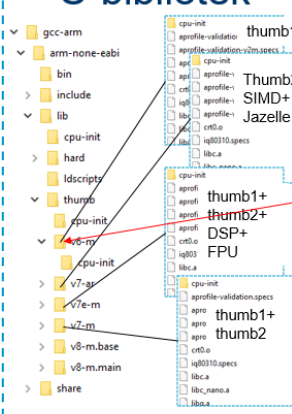
```
/*
 * libMD407.h
 * Declaration of library functions, constants etc...
 */
#include <stdio.h>
#include <errno.h>
#include <sys/stat.h>
/* Type definitions */
typedef struct
{
    ...
    volatile unsigned short sr;
} DEV_DRIVER_DESC, *PDEV_DRIVER_DESC;
/* Constants */
#define MAX_FILENO 5
/* Constant defined by linker */
extern char __heap_low;
extern char __heap_top;
extern char __bss_start;
extern char __bss_end;
/* Library defined functions */

void crt_init(void);
void crt_deinit( void );

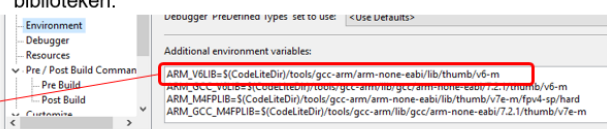
char *_sbrk(int);
int _close(int file);
int _fstat(int file, struct stat *st);
etc ...
```

libMD407.a, installation och användning

C-bibliotek – olika konfigurationer



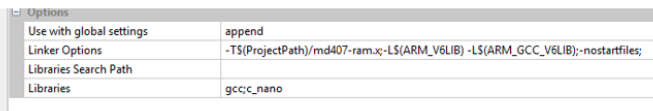
På samma sätt finns det en konfiguration (uppsättning C-bibliotek) för *varje* arkitektur. Vi använder miljövariabler för att precisera biblioteken:



I kommandoraden för länkaren kan vi sedan enkelt välja konfiguration. I listan av bibliotek utelämnar man (konvention) delar av namnet, dvs:

- `libgcc.a` skrivs `gcc`
- `libc_nano.a` skrivs `c_nano`

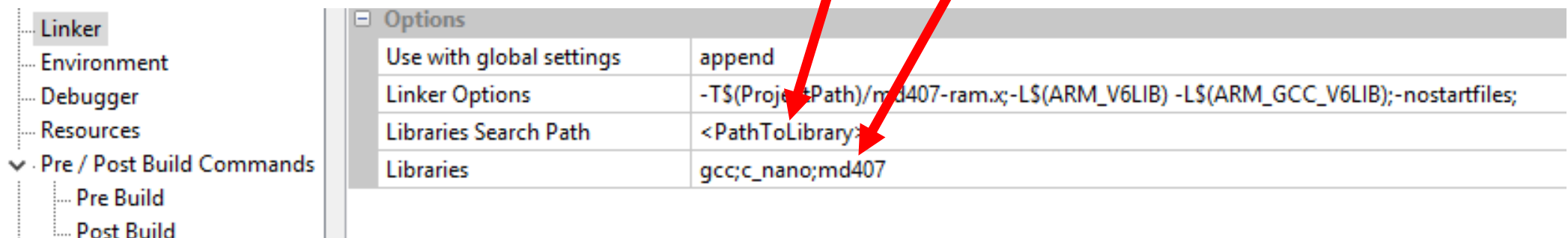
Vi väljer också att använda vår egen "run-time-miljö" (**-nostartfiles**)



Sökväg till det nya biblioteket.

Namn på nytt bibliotek.

Man kan välja att installera biblioteket i en befintlig sökväg, eller att skapa en ny. I vilket fall, måste man lägga till programbiblioteket i listan av bibliotek:



Options	
Use with global settings	append
Linker Options	-T\$(ProjectPath)/md407-ram.x;-L\$(ARM_V6LIB) -L\$(ARM_GCC_V6LIB);-nostartfiles;
Libraries Search Path	<PathToLibrary>
Libraries	gcc;c_nano;md407