

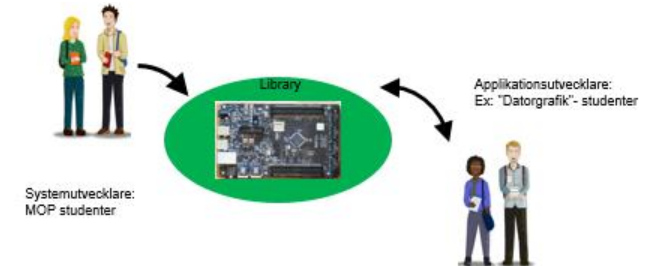
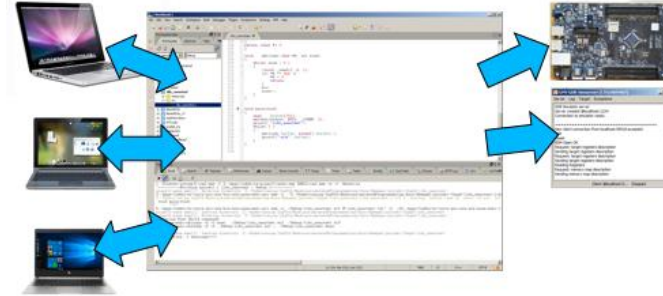
Programbibliotek

Ur innehållet:

Exekveringsmiljö

Olika typer av programbibliotek

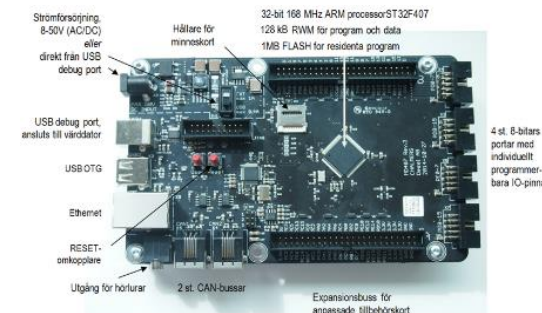
- Kompilatorbibliotek
- C-bibliotek



Målsättningar:

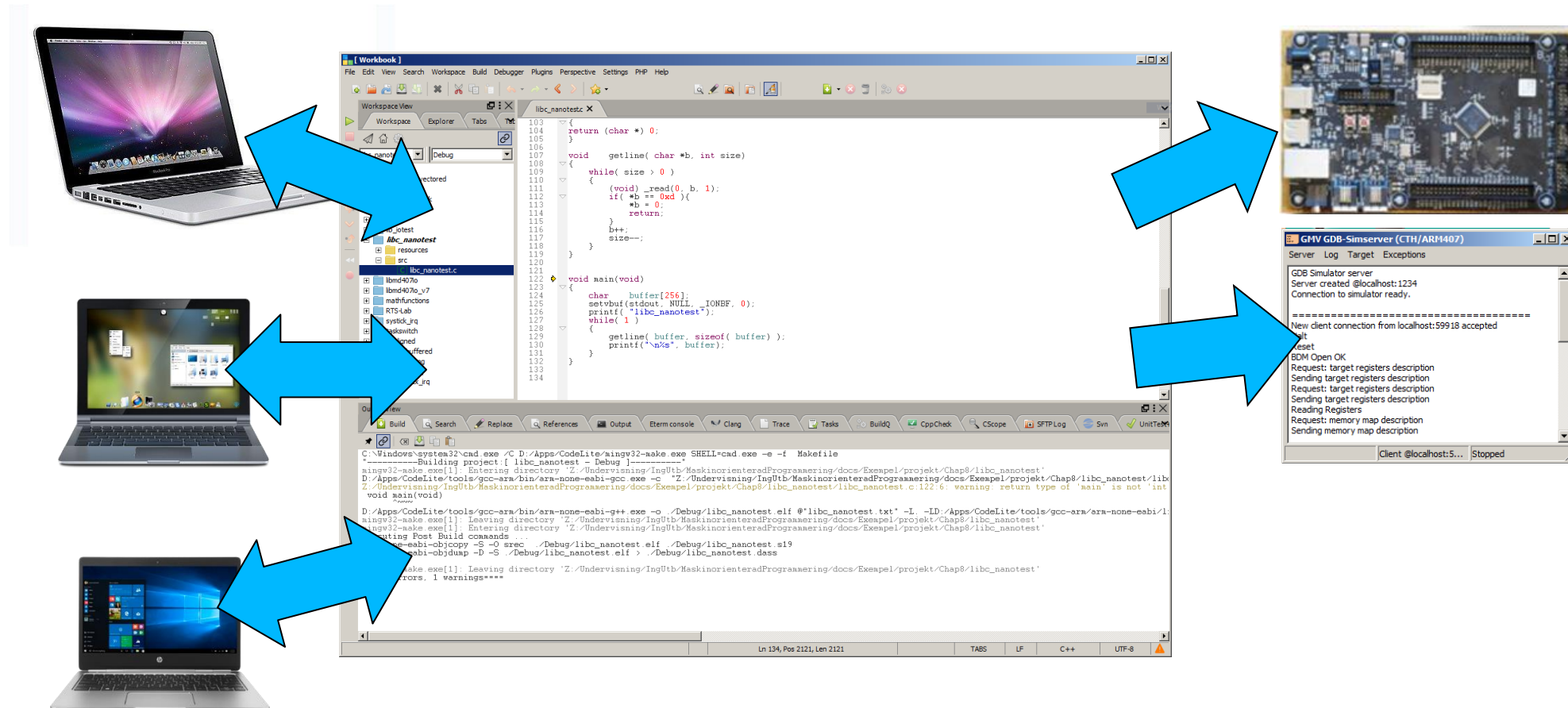
Att kunna använda färdiga programbibliotek

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.



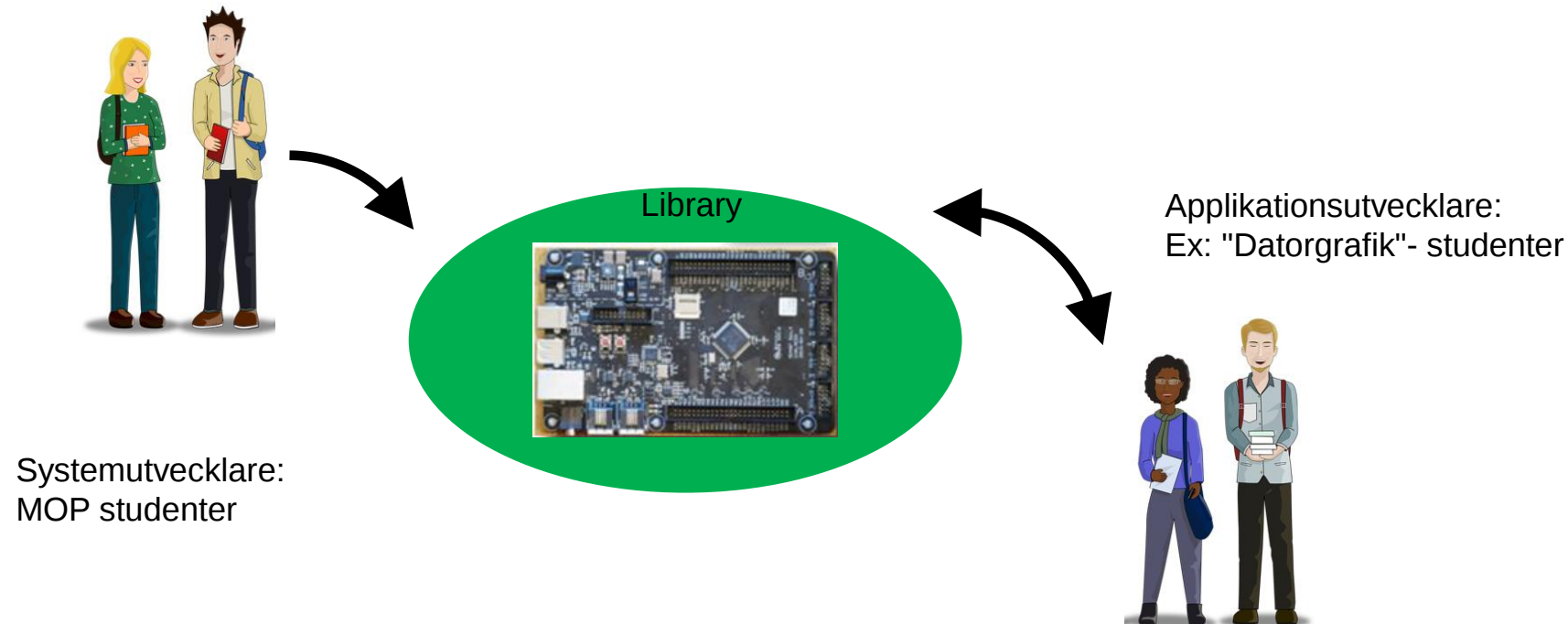
System-/Applikations- programmering för MD407

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.*

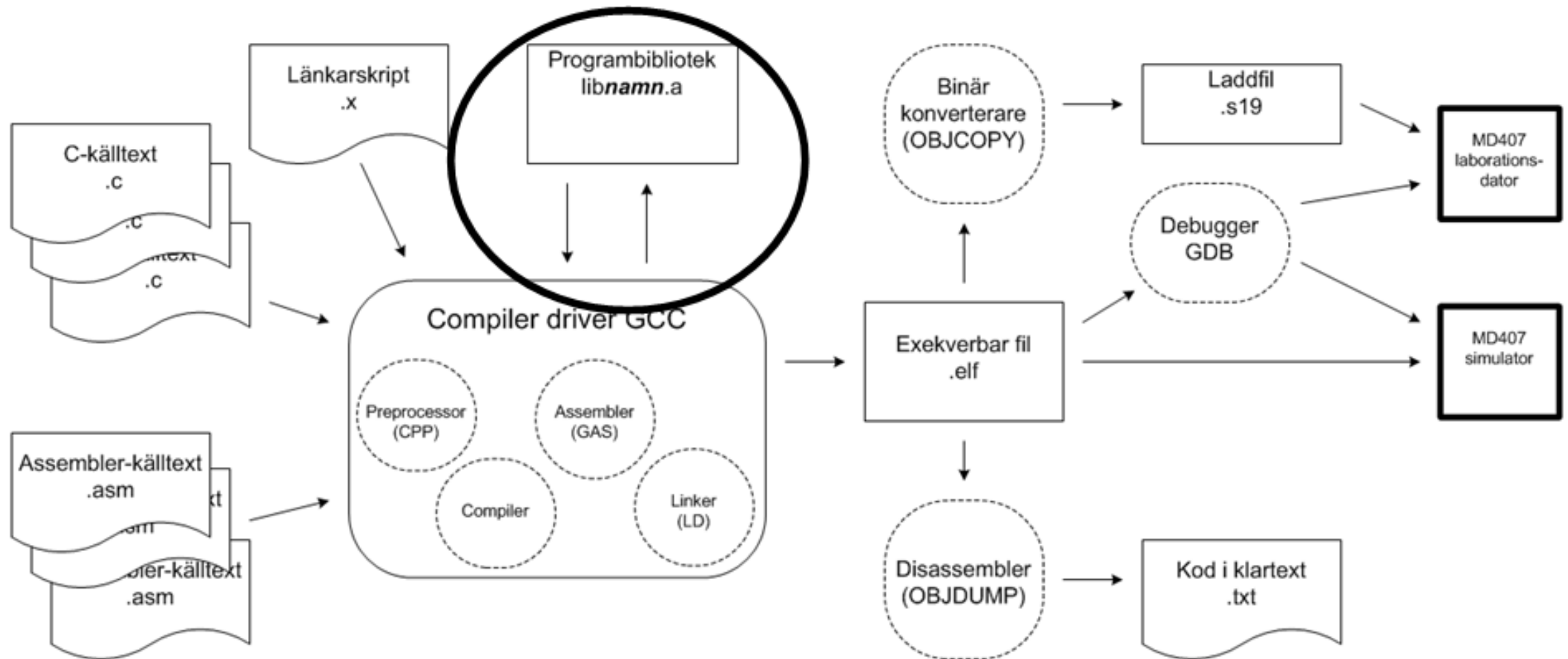
Med *systemprogramvara* menar vi oftast "basal" programvara för vår dator, ett programlager med generellt användbar natur. *Exempel: operativsystem och drivrutiner.*

Med *applikationsprogramvara*, menar vi program avsedda för mer specifik användning.

Exempel: Spel, navigation, övervakning/styrning etc.



Programbibliotek



Programbibliotek – olika typer

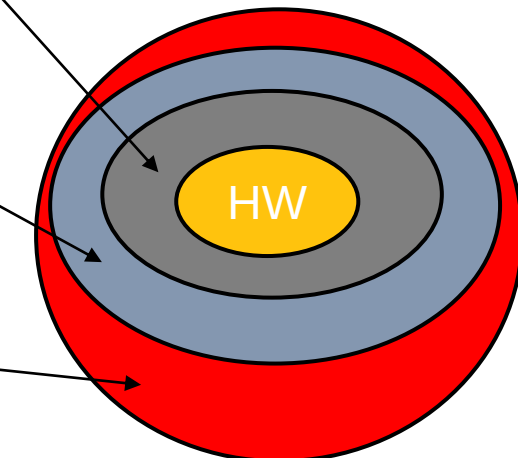
Programbibliotek kan ofta ses som "kompatibilitetslager" där en anpassning sker till någon speciell maskinvara och något specifikt operativsystem.

- 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

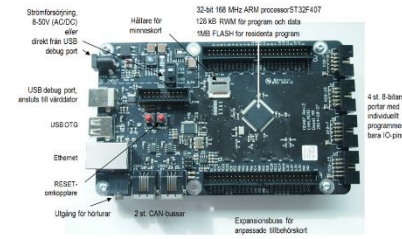
Exempel:
malloc/free, printf...

Exempel: GUI: Fönster
och menyer



Måldator – arkitektur

En måldator kan karakteriseras av:
arkitektur, minnesdisposition och periferienheter.



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

- Thumb 1
- Thumb 2
- DSP (Digital Signal Processing)
- FP (Floating Point)

Instruktion	Storlek	Cortex M0	Cortex M0+	Cortex M1	Cortex M2	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, (STMIA, STMEA), STR, STRB, STRH, SUB, SVC, SXTB, SXTL, 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, LDMD, LDR, LDRB, LDRBT, LDRD, LDREX, LDREXB, LDREXH, LDRH, LDRHT, LDRSB, LDRSHT, LDRSHT, LDRT, MCR, LSL, LSR, MLS, MCRR, MLA, MOV, MOVT, MRC, MRRC, MUL, MVN, NOP, ORN, ORR, PLD, PLDW, PLI, POP, PUSH, RBIT, REV, REV16, REVSH, ROR, RRX, RSB, SBC, SBF, SDIV, SEV, SMLAL, SMULL, SSAT, STC, STMD, STR, STRB, STRBT, STRD, STREX, STREXB, STREXH, STRH, STRHT, STRT, SUB, SXTB, SXTL, TBB, TBH, TEQ, TST, UBF, UDIV, UMLAL, UMULL, USAT, UXTB, UXTH, WFE, WFI, YIELD	32-bit				x	x	x	v7
PKH, QADD, QADD16, QADD8, QASX, QDADD, QDSUB, QSAX, 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, SMLSD, SMLSLD, SMMLA, SMMLS, SMMUL, SMUAD, SMULBB, SMULBT, SMULTT, SMULTB, SMULWT, SMULWB, SMUSD, SSAT16, SSAX, SSUB16, SSUB8, SXTAB, SXTAB16, SXTAH, SXTB16, UADD16, UADD8, UASX, UHADD16, UHADD8, UHASX, UHSAX, UHSUB16, UHSUB8, UMAAL, UQADD16, UQADD8, UQASX, UQSAX, UQSUB16, UQSUB8, USAD8, USADA8, USAT16, USAX, USUB16, USUB8, UXTAB, UXTAB16, UXTAH, UXTB16	32-bit					x	x	v7e (DSP)
VABS, VADD, VCMPE, VCVT, VCVTR, VDIV, VLDM, VLDR, VMLA, VMLS, VMOV, VMRS, VMSR, VMUL, VNEG, VNMLA, VNMLS, VNMUL, VPOP, VPUSH, VSQRT, VSTM, VSTR, VSUB	32-bit					SP FPU	SP FPU	32-b FP
FP- dubbel precision	32-bit						DP FPU	64-b FP

Måldator – minnesdisposition

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

Adressrum	
Tillverkar-specifikt	FFFF FFFF E010 0000 E00F FFFF
Periferibuss	E000 0000 DFFF 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

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

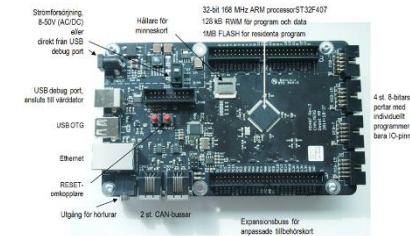
Block 1 SRAM

2001 FFFF
2001 C400
2001 C3FF
2001 C000
2001 BFFF

Monitor/
debugger
stack och data
Relokerade
avbrottsvektorer

Reserverat för
applikationen

2000 0000



Måldator – periferienheter

- Tillgängliga periferienheter –
Systemenheter
IO-enheter

Adressrum

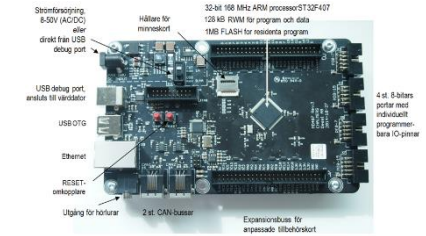
Tillverkar- specifikt	FFFF FFFF	
	E010 0000	
	E00F FFFF	
Periferibuss		
	E000 0000	
	DFFF FFFF	
Externa enheter		
	A000 0000	
	9FFF FFFF	
Extern RAM- minne		
	6000 0000	
	5FFF FFFF	
Periferekretsar		
	4000 0000	
	3FFF FFFF	
Block 1 SRAM		
	2000 0000	
	1FFF FFFF	
Block 0 kod		
	0000 0000	

Periferibuss

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

Periferekretsar

A000 0000 - A000 0FFF	FSMC control reg.	AHB3
5006 0800 - 5006 0BFF	RNG	
5006 0400 - 5006 07FF	HASH	
5006 0000 - 5006 03FF	CRYPT	
5005 0000 - 5005 03FF	DCM	
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	DMA2	
4002 6000 - 4002 63FF	DMA1	
4002 4000 - 4002 4FFF	BMXSRAM	
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	GPIOF	
4002 1000 - 4002 13FF	GPIOE	
4002 0C00 - 4002 0FFF	GPIOD	
4002 0800 - 4002 0BFF	GPIOC	
4002 0400 - 4002 07FF	GPIOB	
4002 0000 - 4002 03FF	GPIOA	
4001 6800 - 4001 6BFF	LCD-TFT	
4001 5800 - 4001 5BFF	SAI1	
4001 5400 - 4001 57FF	SPW	
4001 5000 - 4001 53FF	SPIS	
4001 4800 - 4001 4BFF	TIM11	
4001 4400 - 4001 47FF	TIM10	
4001 4000 - 4001 43FF	TIM9	
4001 3C00 - 4001 3FFF	EXTI	
4001 3800 - 4001 3BFF	SysCFG	
4001 3400 - 4001 37FF	SPM4	
4001 3000 - 4001 33FF	SPM1	
4001 2C00 - 4001 2FFF	SDIO	
4001 2000 - 4001 23FF	ADC1-ADC3-ADC3	
4001 1400 - 4001 17FF	USART6	
4001 1000 - 4001 13FF	USART5	
4001 0400 - 4001 07FF	TIM8	
4001 0000 - 4001 03FF	TIM7	
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	I2C3	
4000 5800 - 4000 5BFF	I2C2	
4000 5400 - 4000 57FF	I2C1	
4000 5000 - 4000 53FF	UART5	
4000 4C00 - 4000 4FFF	UART4	
4000 4800 - 4000 4BFF	USART3	
4000 4400 - 4000 47FF	USART2	
4000 4000 - 4000 43FF	I2SextL	
4000 3C00 - 4000 3FFF	SPM3-SPM3	
4000 3800 - 4000 3BFF	SPM2-SPM2	
4000 3400 - 4000 37FF	I2Sext	
4000 3000 - 4000 33FF	IWDG	
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;  
}
```

gencodeV7.dass (armv7-m)

```
LDR    R2,b  
LDR    R3,c  
SDIV  R3,R2,R3  
LDR    R2,=a  
STR    R3,[R2]
```

gencodev6.dass (armv6-m)

```
LDR    R0,b  
LDR    R1,c  
BL      __divsi3  
LDR    R3,=a  
STR    R0,[R3]
```

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

```
1000101001110001100101  
__divsi3:  
0010110000101000100100100101000  
1010011100011.....
```

Funktionerna i kompilatorbiblioteket används av kompilatorn 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 mjukvaruimplementering av operationerna.

EXEMPEL 4.39 ROBERTSON'S 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
	+00110	
	00011	110
P= 30 =PP(4)=P	0001	1110

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

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.

R(2) *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 $i = 2 \dots n$

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

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 utelämnas denna siffra och på så vis åstadkomma ett kompaktare format.

Exempel: Omvandling av heltal till IEEE flyttal

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

Vi har tidigare kommit fram till resultatet:

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

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:

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

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

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

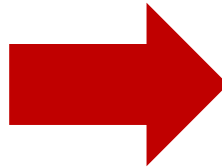
Det innebär exempelvis att tilldelningen:

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

ska göras enligt exemplet ovan...

Kompilatorbibliotek

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



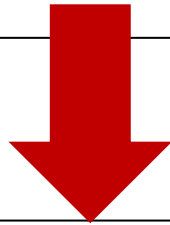
(armv6-m) exempelvis Cortex M3

Flaggor till kompilatorn:

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

ger koden:

```
LDR    R0, i  
BL    __aeabi_i2f  
LDR    R1, =f  
STR    R0, [r1]
```



(armv7-m + floating point unit) exempelvis Cortex M4

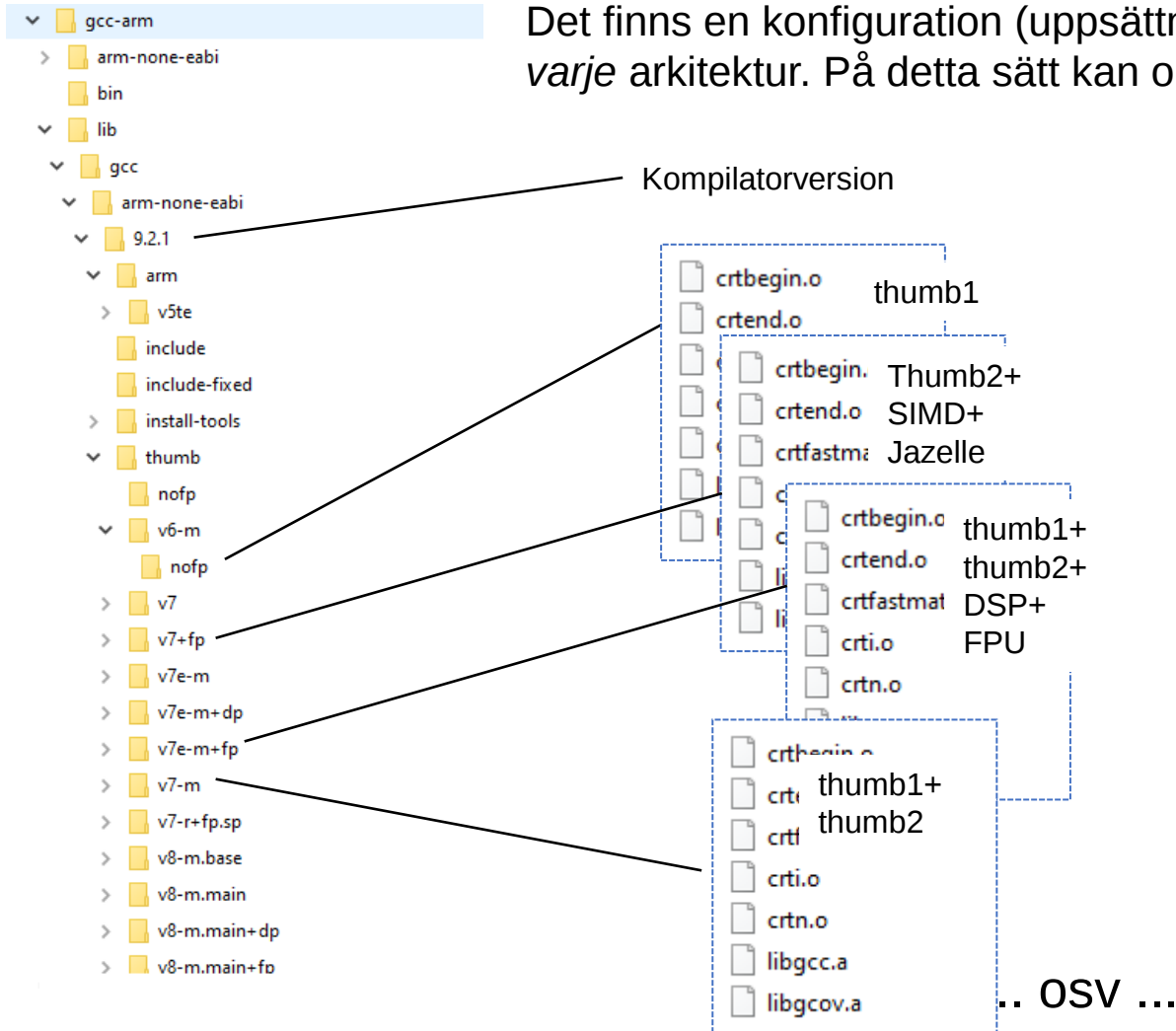
Flaggor till kompilatorn:

```
-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
crtbegin, crtend etc..
Kallas också "startfiles"

Libgcc, kompatibilitetskod

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

"C-nano" - lättviktsimplementering av Standard C

Applikation...

```
#include <stdio.h>

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

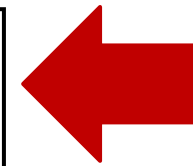
Standard C-bibliotek

libc_nano.a

```
100010100111000110101
printf:
001011000010100010010
010010100010100111000
11.....
```

C-RunTime, (CRT)
funktioner

```
open()
close()
write()
read(),
etc måste utformas baserat
på hårdvaran
```



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, i form av så kallade *drivrutiner*, för MD407.

Exempel: "Device driver" keypad och ASCII-display

Datastruktur: "Device driver".

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

typedef struct
{
    char name[16];
    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
};
```

```
static DEV_DRIVER_DESC AsciiDisplay =
{
    {"AsciiDisplay"},
    asciidisplay_init,
    asciidisplay_deinit,
    0,
    0,
    0,
    0,
    0,
    0,
    asciidisplay_write,
    0
};
```

Exempel: Tabell med "Device drivers"

```
#define MAX_FILENO 5

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

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

```
static DEV_DRIVER_DESC StdIn =
{
    {"stdin"},
    usart_init,
    usart_deinit,
    0,
    usart_isatty,
    0,
    0,
    0,
    0,
    usart_read
};
```

```
static DEV_DRIVER_DESC StdOut =
{
    {"stdin"},
    0,
    usart_deinit,
    0,
    usart_isatty,
    0,
    0,
    0,
    usart_write,
    0
};
```

"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 deklaras 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 teckenorienterade enheter.

_isatty ska returnera 1 om filen är av terminal-typ. stdin, stdout och stderr, kan normalt dirigeras om till blockorienterade enheter, dock inte i vår implementering. isatty implementeras därför, men enbart för stdin, stdout och stderr.

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

"C-run time" - lättviktsimplementering för MD407

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

Ej tillämbart, vi har inget filsystem, våra enheter öppnas (`init`) av runtime-systemet. Det räcker då att vår implementering returnerar en felkod om `open` anropas.

```
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 enheter stängs av runtime-systemet (`deinit`).

```
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 eller enhet.

Funktionen ska dirigera operationen till enhetens write-funktion om en sådan finns.

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

_read, läs från en öppen fil eller enhet.

Funktionen ska dirigera operationen till enhetens read-funktion om en sådan finns.

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

Skapa run-time bibliotek för MD407

Under laboration 6 ska du skapa ett programbibliotek `libmd407.a` som kompletterar med de hårdvaruberoende run-time funktionerna

```
/*
 * libmd407.c
 */
/*
 * driver_asciisp
 * MD407 library
 */
/*
 * driver_keypad
 * MD407 library
 */
/*
 * declarations
 */
#include "libmd407.h"

static int keypad_init(void);
static void keypad_deinit(void);
static int keypad_write(char *ptr, int len);
static int keypad_read(char *ptr, int len);

DEV_DRIVER_DESC StdIn =
{
    {"stdin"},
    keypad_init,
    keypad_deinit,
    keypad_write,
    keypad_read,
};

DEV_DRIVER_DESC StdOut =
{
    ...
};

DEV_DRIVER_DESC StdErr =
{
    ...
};

/* Define functions here */
void usart_init( int initval ) { ... }
void usart_deinit( int deinitval ) { ... }
void usart_isatty( void ) { ... }
int usart_write(char *ptr, int len) { ... }
int usart_read(char *ptr, int len) { ... }
```

```
/*
 * libmd407.h
 * Declaration of library functions, constants etc...
 */
#ifndef _LIBMD407_H
#define _LIBMD407_H
#include <stdio.h>
#include <errno.h>
#include <sys/stat.h>
#include <unistd.h>

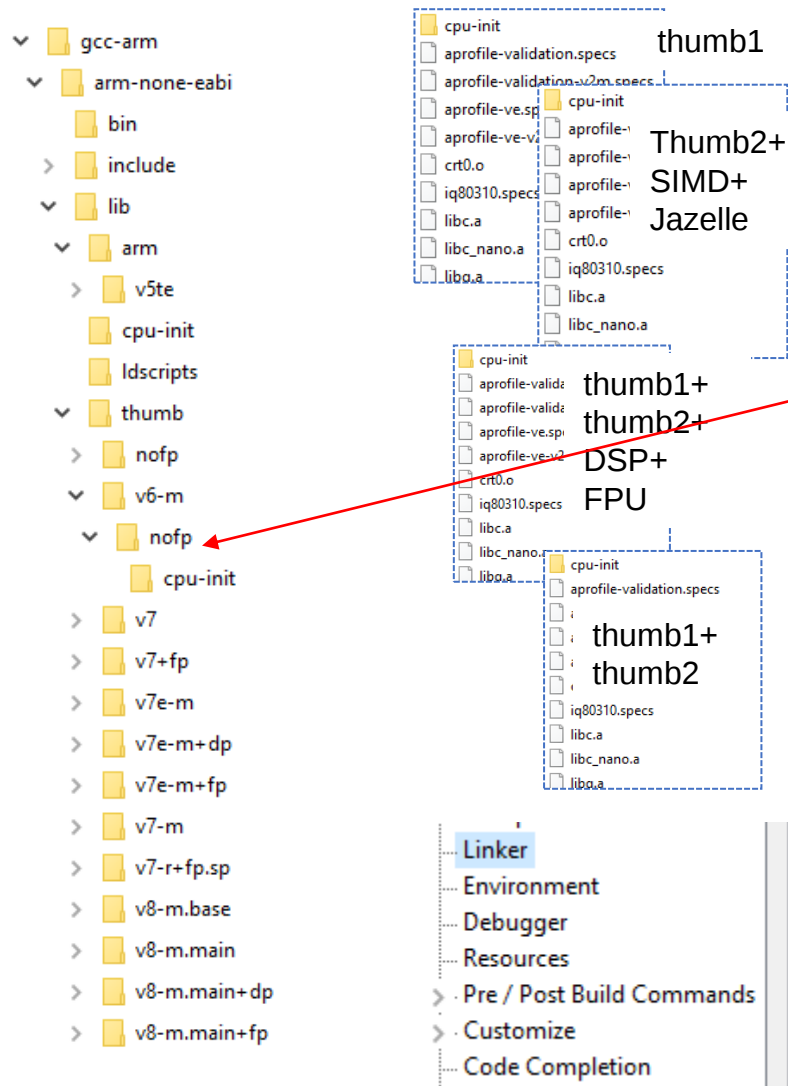
/* Type definitions */
typedef struct
{
    ...
} DEV_DRIVER_DESC, *PDEV_DRIVER_DESC;

/* 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__;

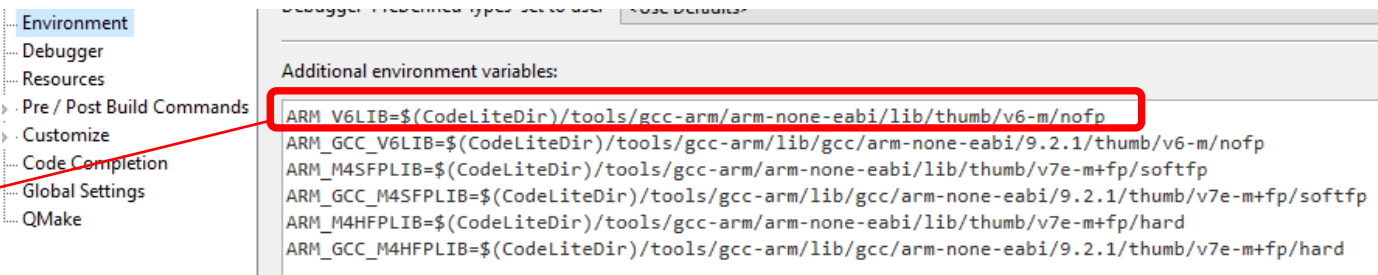
/* Library defined functions */
void crt_init( void );
void crt_deinit( void );

#endif /* LIBMD407_H */
```

C-bibliotek – olika konfigurationer



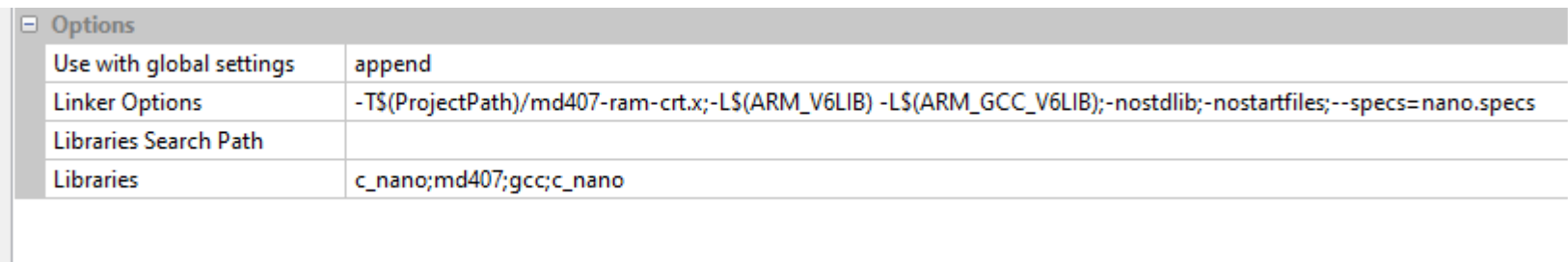
På samma sätt finns det en konfiguration (uppsättning C-bibliotek) för *varje* arkitektur. Vi använder miljövariabler för att precisera biblioteken:



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`



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
```

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

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

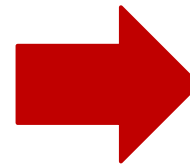
Use with global settings	append
Linker Options	-T\$(ProjectPath)/md407-ram.x;-L\$(ARM_V6LIB) -L\$(ARM_GCC_V6LIB);-nostartfiles;
Libraries Search Path	<PathToLibrary>
Libraries	gcc;c_nano;md407

Demonstration: Användning av biblioteket

Kompilera och installera.....

Testa.....

```
/*  
 * libmd407.c  
 * includes: _crt_init, _crt_deinit, _sbrk for malloc-support.  
 * _exit  
 * template functions for stdio:  
 */  
  
#ifdef USE_REENT  
#define _REENT_SMALL  
#include <sys/reent.h>  
static struct _reent my_clib = _REENT_INIT(my_clib);  
#endif  
/* Include standard headers */  
#include <stdio.h>  
  
/* Declare your functions in 'libmd407.h' */  
#include "libmd407.h"  
  
extern DEV_DRIVER_DESC StdIn, StdOut, StdErr, KeyPad, AsciiDisplay;  
  
/* Order is essential, must match:  
 * enum {STDIN=STDIN_FILENO, STDOUT=STDOUT_FILENO, STDERR=STDERR_FILENO  
 */  
PDEV_DRIVER_DESC device_table[MAX_DEVICE] =  
{  
    {  
        &StdIn,  
        &StdOut,  
        &StdErr,  
        &KeyPad,  
        &AsciiDisplay,  
    },  
};  
  
static char *heap_end;  
char * _sbrk(int incr) {  
    char *prev_heap_end;  
    if (heap_end == 0) {  
        heap_end = malloc(1024);  
    }  
    prev_heap_end = heap_end;  
    if (incr < 0) {  
        heap_end = prev_heap_end - incr;  
    }  
    return heap_end;  
}
```



```
asm volatile(" MOV SP,R0\n");  
asm volatile(" BL _crt_init\n"); /* init C-runtime library */  
asm volatile(" BL main\n"); /* call main */  
asm volatile(" BL _crt_deinit\n"); /* deinit C-runtime library */  
asm volatile(".L1: B .L1\n"); /* never return */  
  
/* Main program */  
void main(void)  
{  
    char c;  
    while(1){  
        (void) read(STDIN_FILENO, &c, 1); /* No buffering here ... */  
        if( c == '\n' ) break;  
        write( ASCIIDISPLAY_FILENO, &c, 1 );  
    }  
}
```