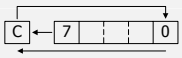
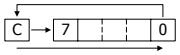
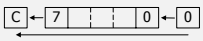
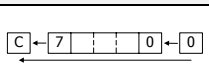
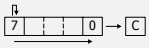
ST7 ASM Quick Reference Guide

A concise listing of the ST7 Instruction set and Directive Language

This quick reference guide gives all the instruction, directives and command line option for the ST7 assembler.

ST7 ASM Instruction Set

Mnemo	Description	Operation	Dest.	Source	Flags	Address Modes
ADC d,s	Add with carry, s to d	d ⇐ d+s+C	A	mem	H,N,Z,C	2,3,4,5,s,w
ADD d,s	Add s to d	d ⇐ d+s	A	mem	H,N,Z,C	2,3,4,5,s,w
AND d,s	Logical AND (d with s)	d ⇐ d AND s	A	mem	N,Z	2,3,4,5,s,w
BCP s,d	Bit compare A, mem	{N,Z}⇐ s AND d	A	mem	N,Z	2,3,4,5,s,w
BRES d,b	Bit reset d	d ⇐ d AND (2**b)	mem			3,4,5,s
BSET d,b	Bit set d	d ⇐ d OR (2**b)	mem			3,4,5,s
BTJF d,b,rel	Jump if bit is false (0)	PC=PC+rel IF (d AND (2**b))=0	mem		C	3,4,5,s
BTJT d,b,rel	Jump if bit is true (1)	PC=PC+rel IF (d AND (2**b))≠0	mem		C	3,4,5,s
CALL d	Call subroutine		mem			3,4,5,s,w
CALLR d	Call subroutine relative		mem			3,5,6,s
CLR d	Clear d	d ⇐ 00	reg,mem		N=0,Z=1	1,3,4,5,s
CP d,s	Arithmetic Compare	{N,Z,C}= Test (d-s)	reg	mem	N,Z,C	2,3,4,5,s,w
CPL d	Logical complement of d	d ⇐ d XOR FF	reg,mem		N,Z,C=1	1,3,4,5,s
DEC d	Decrement d	d⇐d-1	reg,mem		N,Z	1,3,4,5,s
HALT	Halt				I=0	1
INC d	Increment d	d⇐d+1	reg,mem		N,Z	1,3,4,5,s
IRET	Interrupt routine return	Pop CC,A,X,PC			H,I,N,Z,C	1
JP d	Absolute jump	PC ⇐ d	mem			3,4,5,s,w
JRA d	Jump relative always	PC ⇐ PC+d	mem			3,5,6
JRT d	Jump relative if true	PC ⇐ PC+d	mem			3,5,6
JRF d	Never jump	condition false	mem			3,5,6
JRIH d	Jump if Port INT pin = 1	(no port interrupts)	mem			3,5,6
JRIL d	Jump if Port INT pin = 0	(port interrupt)	mem			3,5,6
JRH d	Jump if H = 1	H = 1?	mem			3,5,6
JRNH d	Jump if H = 0	H = 0?	mem			3,5,6
JRM d	Jump if I = 1	I = 1?	mem			3,5,6
JRNM d	Jump if I = 0	I = 0?	mem			3,5,6
JRMI d	Jump if N = 1	N = 1? (minus)	mem			3,5,6
JRPL d	Jump If N = 0	N = 0? (plus)				3,5,6
JREQ d	Jump if Z = 1	Z = 1? (equal)	mem			3,5,6
JRNE d	Jump if Z=0	Z = 0? (not equal)	mem			3,5,6

JRC d	Jump if C=1	C = 1?	mem			3,5,6
JRNC d	Jump if C=0	C = 0?	mem			3,5,6
JRULT d	Jump if C=1	Jump if unsigned $<$	mem			3,5,6
JRUGE d	Jump if C=0	Jump if unsigned $\geq$	mem			3,5,6
JRUGT d	Jump if (C+Z=0)	Jump if unsigned $>$	mem			3,5,6
JRULE d	Jump if (C+Z=1)	Jump if unsigned $\leq$	mem			3,5,6
LD d,s	Load s in d	d $\Leftarrow$ s	reg,mem	mem, reg	N,Z	1,2,3,4,5,s,w
MUL d,s	Multiply d by s	d:s $\Leftarrow$ d*s	A, X, Y	X,Y,A	H=0,C=0	1
NEG d	Negate d (logical 2-complement)	d $\Leftarrow$ (d XOR FF)+1 or 00 - d	reg,mem		N,Z,C	1,3,4,5,s
NOP	No operation					1
OR d,s	OR d with s	d $\Leftarrow$ d OR s	A	mem	N,Z	2,3,4,5,6,s,w
POP d	Pop from the Stack	d $\Leftarrow$ (++SP)	reg, CC		H,I,N,Z,C	1
PUSH d	Push onto the Stack	(SP- -) $\Leftarrow$ d		reg, CC		1
RCF	Reset carry Flag	C = 0			C=0	1
RET	Subroutine return	MSB(PC)=(++SP) LSB(PC)=(++SP)				1
RIM	Reset interrupt mask	I = 0			I=0	1
RLC d	Rotate left through carry		reg,mem		N,Z,C	1,3,4,5,s
RRC d	Rotate right through carry		reg,mem		N,Z,C	1,3,4,5,s
RSP	Reset Stack pointer	SP = Reset Value				1
SBC d,s	Subtract s from d with carry	d $\Leftarrow$ d-s-C	A	mem	N,Z,C	2,3,4,5,s,w
SCF	Set carry flag	C = 1			C=1	1
SIM	Set interrupt mask	I = 1			I=1	1
SLA d	Shift left arithmetic (equal to SLL d=1)		reg,mem		N,Z,C	1,3,4,5,s
SLL d	Shift left logical		reg,mem		N,Z,C	1,3,4,5,s
SRA d	Shift right arithmetic (equal to SLL one)		reg,mem		N,Z,C	1,3,4,5,s
SRL d	Shift right logical		reg,mem		N=0,Z,C	1,3,4,5,s
SUB d,s	Subtract s from d	d $\Leftarrow$ d-s	A	mem	N,Z,C	2,3,4,5,s,w
SWAP d	Swap nibbles	d(7:4) $\Leftrightarrow$ d(3:0)	reg,mem		N,Z	1,3,4,5,s
TNZ d	Test for Neg & Zero	{N,Z}=Test (d)	reg,mem		N,Z	1,3,4,5,s
TRAP	Software trap				I=1	1
WFI	Wait for interrupt				I=0	1
XOR d,s	Exclusive OR (d with s)	d $\Leftarrow$ d XOR s	A	mem	N,Z	2,3,4,5,s,w

Key to ST7 ASM Instruction set

ตัวย่อ / เครื่องหมาย	คำอธิบาย	
d	ปลายทาง (Destination)	
s	ต้นทาง (Source)	
SP	สแต็กพอยน์เตอร์ขนาด 16 บิต (16 bit Stack pointer)	
PC	โปรแกรมเคาน์เตอร์ (Program Counter)	
CC	รีจิสเตอร์คอนดิชันโค้ด (Condition Code Register)	
$\Leftarrow$, $\Leftrightarrow$	ทิศทางการเคลื่อนย้ายข้อมูล (Movement of contents of one location into another)	
ปลายทาง/ต้นทาง/อื่นๆ (Destination / Source / Other)	A	รีจิสเตอร์แอคคิวมูเลเตอร์ (Accumulator register)
	X	รีจิสเตอร์อินเด็กซ์ X (X index register)
	Y	รีจิสเตอร์อินเด็กซ์ Y (Y index register)
	ndx	รีจิสเตอร์อินเด็กซ์ X หรือ Y (X or Y index register)
	reg	รีจิสเตอร์ A, X หรือ Y (A, X or Y register)
	mem	ตำแหน่งแอดเดรสของหน่วยความจำ (Memory location)
	rel	ลาเบลสำหรับการกระโดดแบบสัมพัทธ์ (Relative jump label)
	b	ตำแหน่งบิตภายในรีจิสเตอร์ขนาด 8 บิต (Bit address within an 8 bit file register)
แฟลก (Flag)	H	บิตตัวทดครึ่ง (Half carry bit)
	I	บิตอินเตอร์รัพต์มาสค์ (Interrupt mask bit)
	N	บิตลบ (Negative bit)
	Z	บิตศูนย์ (Zero bit)
	C	บิตตัวทด/ตัวยืม (Carry/Borrow bit)
โหมดการอ้างอิงตำแหน่งแอดเดรส (Addressing modes)	1	ภายใน (Inherent)
	2	ทันทีทันใด (Immediate)
	3	โดยตรง (Direct)
	4	ผ่านค่าอินเด็กซ์ (Indexed)
	5	โดยอ้อม (Indirect)
	6	สัมพัทธ์ (Relative)
	s	แบบใกล้ (Short)
	w	แบบไกล (Long)

ST7 ASM Directive Language Summary

Directive	Description	Syntax
.BELL	Ring bell on console	.BELL
BYTE	Define byte in object code	BYTE <exp or "string">,[,<exp or "string">+]
BYTES	Label type definition: type = byte	BYTES
CEQU	Equate pre-existing label to expression	label CEQU <equ>
.CTRL	Send control codes to the printer	.CTRL <ctrl> [,<ctrl>+]
DATE	Define 12-byte ASCII date into object code	DATE
DC.B	Define byte(s) in object code	DC.B <exp> or "string">, [,<exp or "string">]
DC.W	Define word(s) in object code	DC.W <exp>[,<exp>+]
DC.L	Define long word(s) in object code	DC.L <exp>[,<exp>+]
#DEFINE	Define manifest constant	#DEFINE <CONSTANT ID> <real characters>
DS.B	Define byte space in object code	DS.B [optional number of bytes]
DS.W	Define word space in object code	DS.W [optional number of words]
DS.L	Define long space in object code	DS.L [optional number of long words]
END	End of source code	END
EQU	Equate the label to expression	label EQU <EXPRESSIONS>
EXTERN	Declare external labels	EXTERN
#ELSE	Conditional ELSE	#ELSE
#ENDIF	Conditional terminator	#ENDIF
FCS	Form constant string	FCS <"string"> |<byte> ["string"> |<byte>|]...
.FORM	Set form length of the listing device	.FORM <exp>
GROUP	Name area of source code	GROUP <exp>
#IF	Start conditional assembly	#IF <exp>
#IF1	IF condition that must be satisfied in pass #1 to be true	#IF1
#IF2	IF condition that must be satisfied in pass #2 to be true	#IF2
#IFB	Conditional on argument being blank	#IFB <arg>

#IFIDN	Conditional on arguments being identical	#IFIDN <arg-1> <arg-2>
#IFDEF	Conditional on argument being defined	#IFDEF <exp>
#IFLAB	Conditional on argument being a label	#IFLAB <arg>
#INCLUDE	Insert external source code file	#INCLUDE "filename>"
INTEL	Force Intel-style radix specifier	INTEL
.LALL	List whole body of macro cells	.LALL
.LIST	Enable listing (default)	.LIST
#LOAD	Load named object file at link time	#LOAD "pathname\filename[.ext]"
LOCAL	Define labels as local to macro	LOCAL <arg>
LONG	Define long word in object code	LONG <exp>[,exp>...]
LONGS	Default new label length long	LONGS
MACRO	Define macro template	<macro> MACRO [param-1] [,param-2]...
MEND	End of macro definition	MEND
MOTOROLA	Force Motorola-style radix specifier	MOTOROLA
.NOCHANGE	List original #DEFINE strings	.NOCHANGE
.NOLIST	Turn off listing	.NOLIST
%OUT	Output string ton the console	%OUT string
.PAGE	Perform a form feed	.PAGE
PUBLIC	Make labels public	PUBLIC <arg>
REPEAT	Assembly-time loop initiator	REPEAT
.SALL	Suppress all of the body of called macro	.SALL
SEGMENT	Start of a new segment	|<name>|SEGMENT <align> <combine>'<class>' [cod]
.SETDP	Set base address for direct page	.SETDP <base address>
SKIP	Insert given number of bytes with an initialization value	SKIP <number of bytes>, <value to fill>
STRING	Define a byte-level string	STRING <exp or "string">, [,<exp or "string">...]
SUBTTL	Define a subtitle for listing heading	SUBTTL "<Subtitle string>"
.TAB	Set listing field lengths	.TAB <label>, <opcode>, <operand>, <comment>
TEXAS	Texas Instruments-style radix specifier	TEXAS
TITLE	Define main tit|le for listing	TITLE "<Title string>"|
UNTIL	Assembly time loop terminator	UNTIL <exp>

WORD	Define word in object code	WORD <exp>[, <exp>...]
WORDS	Default new label length word	WORDS
.XALL	List only code producing macro lines	.XALL
ZILOG	Force Zilog-style radix specifiers	ZILOG

ST7 Addressing Mode Syntax

Mode			Syntax	Addressing Mode Range	Ptr Adr	Ptr Size	Length
Inherent			Nop				+ 0
Immediate			ld A,#$55				+ 1
Short	Direct		ld A,$10	00..FF			+ 1
Long	Direct		ld A,$1000	0000..FFFF			+ 2
No offset	Direct	Indexed	ld A,(X)	00..FF			+ 0
Short	Direct	Indexed	ld A,($10,X)	00..1FE			+ 1
Long	Direct	Indexed	ld A,($1000,X)	0000..FFFF			+ 2
Short	Indirect		ld A,[$10]	00..FF	00..FF	byte	+ 2
Long	Indirect		ld A,[$10.w]	0000..FFFF	00..FF	word	+ 2
Short	Indirect	Indexed	ld A,([$10],X)	00..1FE	00..FF	byte	+ 2
Long	Indirect	Indexed	ld A,([$10.w],X)	0000..FFFF	00..FF	word	+ 2
Relative	Direct		jrne loop	PC 127			+ 1
Relative	Indirect		jrne [$10]	PC 127			+ 2
Bit	Direct		bset $10,#7	00..FF			+ 1
Bit	Indirect		bset [$10],#7	00..FF	00..FF	byte	+ 2
Bit	Direct	Relative	btjt $10,#7,skip	00..FF			+ 2
Bit	Indirect	Relative	btjt [$10],#7,skip	00..FF	00..FF	byte	+ 3

ST7 Core Description