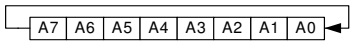
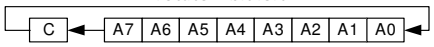
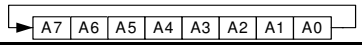
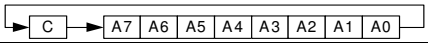
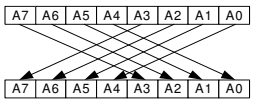
	Mnemonic	Instruction code								Clocks/instruction (42ns/clock)				Explanation
		D7	D6	D5	D4	D3	D2	D1	D0	8051	0 WAIT	1 WAIT	PAR.	
Arithmetic instructions	ADD A,Rn	0	0	1	0	1	n2	n1	n0	12	2	4	2 or 4	(A)<--(A)+(Rn)
	ADD A,direct	0	0	1	0	0	1	0	1	12	2	6	4	(A)<--(A)+(direct)
		a7	a6	a5	a4	a3	a2	a1	a0					
	ADD A,@Ri	0	0	1	0	0	1	1	i	12	2	4	2 or 4	(A)<--(A)+((Ri))
	ADD A,#data	0	0	1	0	0	1	0	0	12	2	6	4	(A)<--(A)+#data
		d7	d6	d5	d4	d3	d2	d1	d0					
	ADDC A,Rn	0	0	1	1	1	n2	n1	n0	12	2	4	2 or 4	(A)<--(A)+(Rn)+(C)
	ADDC A,direct	0	0	1	1	0	1	0	1	12	2	6	4	(A)<--(A)+(direct)+(C)
		a7	a6	a5	a4	a3	a2	a1	a0					
	ADDC A,@Ri	0	0	1	1	0	1	1	i	12	2	4	2 or 4	(A)<--(A)+((Ri))+(C)
	ADDC A,#data	0	0	1	1	0	1	0	0	12	2	6	4	(A)<--(A)+#data+(C)
		d7	d6	d5	d4	d3	d2	d1	d0					
	SUBB A,Rn	1	0	0	1	1	n2	n1	n0	12	2	4	2 or 4	(A)<--(A)-(Rn)-(C)
	SUBB A,direct	1	0	0	1	0	1	0	1	12	2	6	4	(A)<--(A)-(direct)-(C)
		a7	a6	a5	a4	a3	a2	a1	a0					
	SUBB A,@Ri	1	0	0	1	0	1	1	i	12	2	4	2 or 4	(A)<--(A)-((Ri))-(C)
	SUBB A,#data	1	0	0	1	0	1	0	0	12	2	6	4	(A)<--(A)-#data-(C)
		d7	d6	d5	d4	d3	d2	d1	d0					
	INC A	0	0	0	0	0	1	0	0	12	2	4	2 or 4	(A)<--(A)+1
	INC Rn	0	0	0	0	1	n2	n1	n0	12	2	4	2 or 4	(Rn)<--(Rn)+1
	INC direct	0	0	0	0	0	1	0	1	12	2	6	4	(direct)<--(direct)+1
		a7	a6	a5	a4	a3	a2	a1	a0					
	INC @Ri	0	0	0	0	0	1	1	i	12	2	4	2 or 4	((Ri))<--((Ri))+1
	INC DPTR	1	0	1	0	0	0	1	1	24	4	4	4	(DPTR)<--(DPTR)+1
	DEC A	0	0	0	1	0	1	0	0	12	2	4	2 or 4	(A)<--(A)-1
	DEC Rn	0	0	0	1	1	n2	n1	n0	12	2	4	2 or 4	(Rn)<--(Rn)-1
	DEC direct	0	0	0	1	0	1	0	1	12	2	6	4	(direct)<--(direct)-1
		a7	a6	a5	a4	a3	a2	a1	a0					
	DEC @Ri	0	0	0	1	0	1	1	i	12	2	4	2 or 4	((Ri))<--((Ri))-1
	MUL AB	1	0	1	0	0	1	0	0	96	8	8	8	(B),(A)<--(A)x(B) B<--MSB A<--LSB
	DIV AB	1	0	0	0	0	1	0	0	96	8	8	8	(A),(B)<--(A)/(B) A<--Q B<--R
	DA A	1	1	0	1	0	1	0	0	12	2	4	2 or 4	Contents A=BCD , use ADD or ADDC with BCD If flags not modified DA A will adjust result to BCD
Logical instructions	ANL A,Rn	0	1	0	1	1	n2	n1	n0	12	2	4	2 or 4	(A)<--(A) AND (Rn)
	ANL A,direct	0	1	0	1	0	1	0	1	12	2	6	4	(A)<--(A) AND (direct)
		a7	a6	a5	a4	a3	a2	a1	a0					
	ANL A,@Ri	0	1	0	1	0	1	1	i	12	2	4	2 or 4	(A)<--(A) AND ((Ri))
	ANL A,#data	0	1	0	1	0	1	0	0	12	2	6	4	(A)<--(A) AND #data
		d7	d6	d5	d4	d3	d2	d1	d0					
	ANL direct,A	0	1	0	1	0	0	1	0	12	2	6	4	(direct)<--(direct) AND (A)
		a7	a6	a5	a4	a3	a2	a1	a0					
	ANL direct,#data	0	1	0	1	0	0	1	1	24	4	10	6 or 8	(direct)<--(direct) AND #data
		a7	a6	a5	a4	a3	a2	a1	a0					
		d7	d6	d5	d4	d3	d2	d1	d0					
	ORL A,Rn	0	1	0	0	1	n2	n1	n0	12	2	4	2 or 4	(A)<--(A) OR (Rn)
	ORL A,direct	0	1	0	0	0	1	0	1	12	2	6	4	(A)<--(A) OR (direct)
		a7	a6	a5	a4	a3	a2	a1	a0					

	Mnemonic	Instruction code								Clocks/instruction (42ns/clock)				Explanation
		D7	D6	D5	D4	D3	D2	D1	D0	8051	0 WAIT	1 WAIT	PAR.	
Logical instructions	ORL A,@Ri	0	1	0	0	0	1	1	i	12	2	4	2 or 4	(A)<--(A) OR ((Ri))
	ORL A,#data	0 d7	1 d6	0 d5	0 d4	0 d3	1 d2	0 d1	0 d0	12	2	6	4	(A)<--(A) OR #data
	ORL direct,A	0 a7	1 a6	0 a5	0 a4	0 a3	0 a2	1 a1	0 a0	12	2	6	4	(direct)<--(direct) OR (A)
	ORL direct,#data	0 a7 d7	1 a6 d6	0 a5 d5	0 a4 d4	0 a3 d3	0 a2 d2	1 a1 d1	1 a0 d0	24	4	10	6 or 8	(direct)<--(direct) OR #data
	XRL A,Rn	0	1	1	0	1	n2	n1	n0	12	2	4	2 or 4	(A)<--(A) XOR (Rn)
	XRL A,direct	0 a7	1 a6	1 a5	0 a4	0 a3	1 a2	0 a1	1 a0	12	2	6	4	(A)<--(A) XOR (direct)
	XRL A,@Ri	0	1	1	0	0	1	1	i	12	2	4	2 or 4	(A)<--(A) XOR ((Ri))
	XRL A,#data	0 d7	1 d6	1 d5	0 d4	0 d3	1 d2	0 d1	0 d0	12	2	6	4	(A)<--(A) XOR #data
	XRL direct,A	0 a7	1 a6	1 a5	0 a4	0 a3	0 a2	1 a1	0 a0	12	2	6	4	(direct)<--(direct) XOR (A)
	XRL direct,#data	0 a7 d7	1 a6 d6	1 a5 d5	0 a4 d4	0 a3 d3	0 a2 d2	1 a1 d1	1 a0 d0	24	4	10	6 or 8	(direct)<--(direct) XOR #data
	CLR A	1	1	1	0	0	1	0	0	12	2	4	2 or 4	(A)<--0
	CPL A	1	1	1	1	0	1	0	0	12	2	4	2 or 4	ONES COMPLEMENT OF (A)
	RL A	0	0	1	0	0	0	1	1	12	2	4	2 or 4	rotate 1 bit left 
	RLC A	0	0	1	1	0	0	1	1	12	2	4	2 or 4	rotate 1 bit left 
	RR A	0	0	0	0	0	0	1	1	12	2	4	2 or 4	rotate 1 bit right 
	RRC A	0	0	0	1	0	0	1	1	12	2	4	2 or 4	rotate 1 bit right 
	SWAP A	1	1	0	0	0	1	0	0	12	2	4	2 or 4	
Data transfer instructions	MOV A,Rn	1	1	1	0	1	n2	n1	n0	12	2	4	2 or 4	(A)<--(Rn)
	MOV A,direct	1 a7	1 a6	1 a5	0 a4	0 a3	1 a2	0 a1	1 a0	12	2	6	4	(A)<--(direct)
	MOV A,@Ri	1	1	1	0	0	1	1	i	12	2	4	2 or 4	(A)<--((Ri))
	MOV A,#data	0 d7	1 d6	1 d5	1 d4	0 d3	1 d2	0 d1	0 d0	12	2	6	4	(A)<--#data
	MOV Rn,A	1	1	1	1	1	n2	n1	n0	12	2	4	2 or 4	(Rn)<--(A)
	MOV Rn,direct	1 a7	0 a6	1 a5	0 a4	1 a3	n2 a2	n1 a1	n0 a0	24	4	8	6	(Rn)<--(direct)
	MOV Rn,#data	0 d7	1 d6	1 d5	1 d4	1 d3	n2 d2	n1 d1	n0 d0	12	2	6	4	(Rn)<--#data
	MOV direct,A	1 a7	1 a6	1 a5	1 a4	0 a3	1 a2	0 a1	1 a0	12	2	6	4	(direct)<--(A)
	MOV direct,Rn	1 a7	0 a6	0 a5	0 a4	1 a3	n2 a2	n1 a1	n0 a0	24	4	8	6	(direct)<--(Rn)

	Mnemonic	Instruction code								Clocks/instruction (42ns/clock)				Explanation
		D7	D6	D5	D4	D3	D2	D1	D0	8051	0 WAIT	1 WAIT	PAR.	
Data transfer instructions	MOV direct1,direct2	1	0	0	0	0	1	0	1	24	4	10	6 or 8	(direct1)<--(direct2)
		2a7	a6	a5	a4	a3	a2	a1	a0					
		1a7	a6	a5	a4	a3	a2	a1	a0					
	MOV direct,@Ri	1	0	0	0	0	1	1	i	24	4	8	6	(direct)<--((Ri))
		a7	a6	a5	a4	a3	a2	a1	a0					
	MOV direct,#data	0	1	1	1	0	1	0	1	24	4	10	6 or 8	(direct)<--#data
		a7	a6	a5	a4	a3	a2	a1	a0					
		d7	d6	d5	d4	d3	d2	d1	d0					
	MOV @Ri,A	1	1	1	1	0	1	1	i	12	2	4	2 or 4	((Ri))<--(A)
	MOV @Ri,direct	1	0	1	0	0	1	1	i	24	4	8	6	((Ri))<--(direct)
		a7	a6	a5	a4	a3	a2	a1	a0					
	MOV @Ri,#data	0	1	1	1	0	1	1	i	12	2	6	4	((Ri))<--#data
		d7	d6	d5	d4	d3	d2	d1	d0					
	MOV DPTR,#data	1	0	0	1	0	0	0	0	24	4	10	6 or 8	(DPTR)<--#data16
		d15	d14	d13	d12	d11	d10	d9	d8					or: (DPH)<--high #data16 and
		d7	d6	d5	d4	d3	d2	d1	d0					(DPL)<--low #data16
	MOVC A,@A+DPTR	1	0	0	1	0	0	1	1	24	4	6	4|6|8	(A)<--((A)+(DPTR)) **FLASH**
	MOVC @(DPTR++),A	1	0	1	0	0	1	0	1	**ERROR**	4	4	4 or 6	((DPTR))<--(A) and DPTR+1 **FLASH DO NOT USE**
	MOVC A,@A+PC	1	0	0	0	0	0	1	1	24	4	6	4|6|8	(A)<--((A)+(PC+1)) **FLASH**
	MOVX A,@Ri	1	1	1	0	0	0	1	i	24	4	6	4 or 6	(A)<--((Ri)) XRAM (XADDRH<--f0h-f5h)
	MOVX A,@DPTR	1	1	1	0	0	0	0	0	24	4	6	4 or 6	(A)<--((DPTR)) **XRAM**
	MOVX @Ri,A	1	1	1	1	0	0	1	i	24	4	6	4 or 6	((Ri))<--(A) XRAM (XADDRH<--f0h-f5h)
	MOVX @DPTR,A	1	1	1	1	0	0	0	0	24	4	6	4 or 6	((DPTR))<--(A) **XRAM**
	PUSH direct	1	1	0	0	0	0	0	0	24	4	8	6	(SP)<--(SP)+1
		a7	a6	a5	a4	a3	a2	a1	a0					((SP))<--(direct)
	POP direct	1	1	0	1	0	0	0	0	24	4	8	6	(direct)<--((SP))
		a7	a6	a5	a4	a3	a2	a1	a0					(SP)<--(SP)-1
	XCH A,Rn	1	1	0	0	1	n2	n1	n0	12	2	4	2 or 4	(A)<>(Rn)
	XCH A,direct	1	1	0	0	0	1	0	1	12	2	6	4	(A)<>(direct)
		a7	a6	a5	a4	a3	a2	a1	a0					
	XCH A,@Ri	1	1	0	0	0	1	1	i	12	2	4	2 or 4	(A)<>((Ri))
	XCHD A,@Ri	1	1	0	1	0	1	1	i	12	2	4	2 or 4	(A)<>((Ri)) (low nibble only)
Boolean manipulation instructions	CLR C	1	1	0	0	0	0	1	1	12	2	4	2 or 4	(C)<--0
	CLR bit	1	1	0	0	0	0	1	0	12	2	6	4	(bit)<--0
		b7	b6	b5	b4	b3	b2	b1	b0					
	SETB C	1	1	0	1	0	0	1	1	12	2	4	2 or 4	(C)<--1
	SETB bit	1	1	0	1	0	0	1	0	12	2	6	4	(bit)<--1
		b7	b6	b5	b4	b3	b2	b1	b0					
	CPL C	1	0	1	1	0	0	1	1	12	2	4	2 or 4	complement carry flag
	CPL bit	1	0	1	1	0	0	1	0	12	2	6	4	complement (bit)
		b7	b6	b5	b4	b3	b2	b1	b0					
	ANL C,bit	1	0	0	0	0	0	1	0	24	4	8	6	(C)<--(C) AND (bit)
		b7	b6	b5	b4	b3	b2	b1	b0					
	ANL C,/bit	1	0	1	1	0	0	0	0	24	4	8	6	(C)<--(C) AND NOT(BIT)
		b7	b6	b5	b4	b3	b2	b1	b0					
	ORL C,bit	0	1	1	1	0	0	1	0	24	4	8	6	(C)<--(C) OR (bit)
		b7	b6	b5	b4	b3	b2	b1	b0					

	Mnemonic	Instruction code								Clocks/instruction (42ns/clock)				Explanation
		D7	D6	D5	D4	D3	D2	D1	D0	8051	0 WAIT	1 WAIT	PAR.	
Boolean manipulation i.	ORL C,/BIT	1 b7	0 b6	1 b5	0 b4	0 b3	0 b2	0 b1	0 b0	24	4	8	6	(C)<--(C) OR NOT (bit)
	MOV C,bit	1 b7	0 b6	1 b5	0 b4	0 b3	0 b2	1 b1	0 b0	12	2	6	4	(C)<--(bit)
	MOV bit,C	1 b7	0 b6	0 b5	1 b4	0 b3	0 b2	1 b1	0 b0	24	4	8	6	(bit)<--(C)
Program branching	ACALL addr11	a10 a7	a9 a6	a8 a5	1 a4	0 a3	0 a2	0 a1	1 a0	24	4	8	6 or 8	call subroutine (2KByte page) save return address on stack
	LCALL addr16	0 a15 a7	0 a14 a6	0 a13 a5	1 a12 a4	0 a11 a3	0 a10 a2	1 a9 a1	0 a8 a0	24	4	10	8	call subroutine (64KByte range) save return address on stack
	RET	0	0	1	0	0	0	1	0	24	4	4	4 or 6	pop return address and jump
	RETI	0	0	1	1	0	0	1	0	24	4	4	4 or 6	pop return address and jump restore interrupt scanning
	AJMP addr11	a10 a7	a9 a6	a8 a5	0 a4	0 a3	0 a2	0 a1	1 a0	24	4	8	6 or 8	jump to address (2KByte page)
	LJMP addr16	0 a15 a7	0 a14 a6	0 a13 a5	0 a12 a4	0 a11 a3	0 a10 a2	1 a9 a1	0 a8 a0	24	4	10	8	jump to address (64KByte range)
	SJMP rel	1 r7	0 r6	0 r5	0 r4	0 r3	0 r2	0 r1	0 r0	24	4	8	6 or 8	jump rel
	JMP @A+DPTR	0	1	1	1	0	0	1	1	24	4	4	4 or 6	(PC)<--(A)+(DPTR)
	JZ rel	0 r7	1 r6	1 r5	0 r4	0 r3	0 r2	0 r1	0 r0	24	4	8	6 or 8	If (A)<--0 jump rel
	JNZ rel	0 r7	1 r6	1 r5	1 r4	0 r3	0 r2	0 r1	0 r0	24	4	8	6 or 8	If (A) ≠0 jump rel
	JC rel	0 r7	1 r6	0 r5	0 r4	0 r3	0 r2	0 r1	0 r0	24	4	8	6 or 8	If (C)<--1 jump rel
	JNC rel	0 r7	1 r6	0 r5	1 r4	0 r3	0 r2	0 r1	0 r0	24	4	8	6 or 8	If (C)<--0 jump rel
	JB bit,rel	0 b7 r7	0 b6 r6	1 b5 r5	0 b4 r4	0 b3 r3	0 b2 r2	0 b1 r1	0 b0 r0	24	4	10	6 or 8	If (bit)<--1 jump rel
	JNB bit,rel	0 b7 r7	0 b6 r6	1 b5 r5	1 b4 r4	0 b3 r3	0 b2 r2	0 b1 r1	0 b0 r0	24	4	10	6 or 8	If (bit)<--0 jump rel
	JBC bit,rel	0 b7 r7	0 b6 r6	0 b5 r5	1 b4 r4	0 b3 r3	0 b2 r2	0 b1 r1	0 b0 r0	24	4	10	6 or 8	If (bit)<--1 jump rel and clear (bit)
	CJNE A,direct,rel	1 a7 r7	0 a6 r6	1 a5 r5	1 a4 r4	0 a3 r3	1 a2 r2	0 a1 r1	1 a0 r0	24	4	10	6 or 8	If (A)≠(direct) jump rel If (A)<direct (C)<--1
	CJNE A,#data,rel	1 d7 r7	0 d6 r6	1 d5 r5	1 d4 r4	0 d3 r3	1 d2 r2	0 d1 r1	0 d0 r0	24	4	10	6 or 8	If (A)≠ #data jump rel If (A)<#data (C)<--1
	CJNE Rn,#data,rel	1 d7 r7	0 d6 r6	1 d5 r5	1 d4 r4	1 d3 r3	n2 d2 r2	n1 d1 r1	n0 d0 r0	24	4	10	6 or 8	If (Rn)≠ #data jump rel If (Rn)<#data (C)<--1

	Mnemonic	Instruction code								Clocks/instruction (42ns/clock)				Explanation
		D7	D6	D5	D4	D3	D2	D1	D0	8051	0 WAIT	1 WAIT	PAR.	
Program branching	CJNE @Ri,#data,rel	1	0	1	1	0	1	1	i	24	4	10	6 or 8	If ((Ri))≠ #data jump rel
		d7	d6	d5	d4	d3	d2	d1	d0					If ((Ri))<#data (C)<--1
		r7	r6	r5	r4	r3	r2	r1	r0					
	DJNZ Rn,rel	1	1	0	1	1	n2	n1	n0	24	4	8	6 or 8	(Rn)<--(Rn)-1
		r7	r6	r5	r4	r3	r2	r1	r0					If (Rn)≠ 0 jump rel
	DJNZ direct,rel	1	1	0	1	0	1	0	1	24	4	10	6 or 8	(direct)<--(direct)-1
		a7	a6	a5	a4	a3	a2	a1	a0					If (direct)≠ 0 jump rel
		r7	r6	r5	r4	r3	r2	r1	r0					
	NOP	0	0	0	0	0	0	0	0	12	2	4	2 or 4	No OPeration (does nothing)

PAR: parallel read flash is default memory access method (disable: LCALL DFFCh, enable LCALL DFFFh build in routines)

Notes on instruction set and the adressing modes

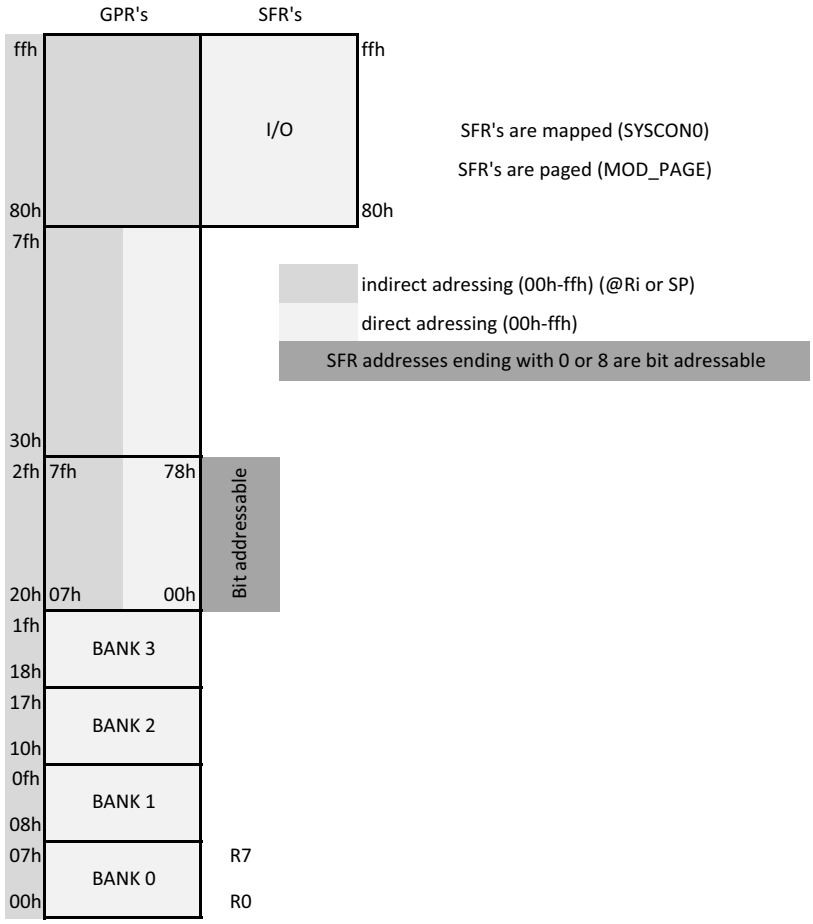
Rn	Register R7-R0 of currently selected register bank (RS0 and RS1 in PSW SFR)
direct	8 bit GPR address (00h-7fh and 80h-ffh SFR register space)
@Ri	Ri is a 8 bit pointer to indirect addressable GPR's (00h-ffh) (i=0 or i=1)
#data	8 bit constant included in instruction (range 00h-ffh)
#data16	16 bit constant included in instruction (range 0000h-ffffh)
addr16	16 bit destination address (range 64KByte)
addr11	11 bit destination address (range within current 2Kbyte page)
rel	8 bit two's complement jump offset (relative to first byte next instruction)
bit	8 bit address of a direct adressable bit

Instructions that affec flag settings

Instruction	Flag		
	C	OV	AC
ADD	x	x	x
ADDC	x	x	x
SUBB	x	x	x
MUL	0	x	
DIV	0	x	
DA A	x		
RRC	x		
RLC	x		
SETB C	1		
CLR C	0		
CPL C	x		
ANL C,bit	x		
ANL C,/bit	x		
ORL C,bit	x		
ORL C,/bit	x		
MOV C,bit	x		
CJNE	x		

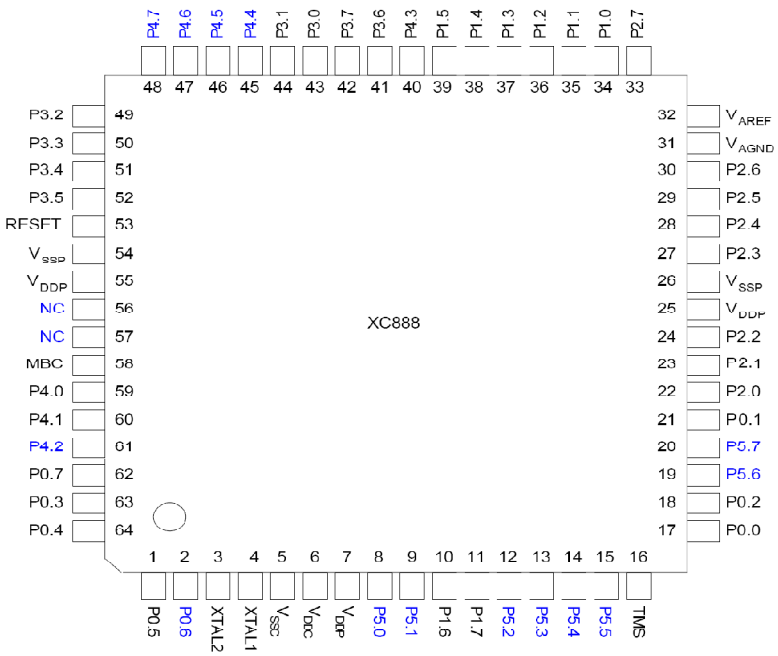
XRAM: f000h-f5ffh
Boot ROM: c000h-efffh
FLASH: 0000h-7fffh

Register map



SFR ADDR	NOT MAPPED (RMAP=0)							RMAP=1
BIT access	PAGE 0	PAGE 1	PAGE 2	PAGE 3	PAGE 4	PAGE 5	PAGE 6	NO PAGE
FF	CCU6_CC62SRH	CCU6_CC62RH	CCU6_TRPCTRH	CCU6_CMPSTATH				
FE	CCU6_CC62SRL	CCU6_CC62RL	CCU6_TRPCTRL	CCU6_CMPSTATL				
FD	CCU6_CC61SRH	CCU6_CC61RH	CCU6_MODCTRH	CCU6_T13H				
FC	CCU6_CC61SRL	CCU6_CC61RL	CCU6_MODCTRL	CCU6_T13L				
FB	CCU6_CC60SRH	CCU6_CC60RH	CCU6_TCTR2H	CCU6_T12H				
FA	CCU6_CC60SRL	CCU6_CC60RL	CCU6_TCTR2L	CCU6_T12L				
F9	IPH1							
F8	IP1							
F7								HWBPDR
F6								HWBPSR
F5								MMDR
F4								MMICR
F3								MMBPCR
F2								MMSR
F1								MMCR2
F0	B							
EF								
EE								
ED								
EC								MMWR2
EB	FDRES							MMWR1
EA	FDSTEP							
E9	FDCON	MISC_CON						MMCR2
E8	IEN1							
E7								
E6								
E5								
E4								
E3								
E2								
E1								
E0	ACC (A ONLY IF USED IN OPCODE)							
DF								
DE	DATA3							
DD	DATA2							
DC	DATA1							
DB	DATA0							
DA	ADH							
D9	ADL							
D8	ADCON							
D7								
D6								
D5								
D4								
D3		ADC_CHCTR7	ADC_RESR3H	ADC_RESRA3H		ADC_EVINPR	ADC_QINR0	
D2		ADC_CHCTR6	ADC_RESR3L	ADC_RESRA3L		ADC_EVINSR	ADC_QBUR0	
D1	ADC_PAGE							
D0	PSW							
CF	ADC_ETRCR	ADC_CHCTR5	ADC_RESR2H	ADC_RESRA2H		ADC_EVINCR	ADC_Q0R0	
CE	ADC_INPCR0	ADC_CHCTR4	ADC_RESR2L	ADC_RESRA2L	ADC_VFCR	ADC_EVINFR	ADC_QSR0	
CD	ADC_LCBR	ADC_CHCTR3	ADC_RESR1H	ADC_RESRA1H	ADC_RCR3	ADC_CHINPR	ADC_QMR0	FDRES1
CC	ADC_PRAR	ADC_CHCTR2	ADC_RESR1L	ADC_RESRA1L	ADC_RCR2	ADC_CHINSR	ADC_CRMR1	FDSTEP1
CB	ADC_GLOBSTR	ADC_CHCTR1	ADC_RESR0H	ADC_RESRA0H	ADC_RCR1	ADC_CHINCR	ADC_CRPR1	FDCON1
CA	ADC_GLOBCTR	ADC_CHCTR0	ADC_RESR0L	ADC_RESRA0L	ADC_RCR0	ADC_CHINFR	ADC_CRCR1	BG1
C9	P4_DIR	P4_PUDEN	P4_ALTSEL1					SBUF1
C8	P4_DATA	P4_PUDSEL	P4_ALTSEL0	P4_OD				SCON1
C7								
C6								
C5	T2_T2H							T21_T2H
C4	T2_T2L							T21_T2L
C3	T2_RC2H							T21_RC2H
C2	T2_RC2L							T21_RCL2
C1	T2_T2MOD							T21_T2MOD
C0	T2_T2CON							T21_T2CON
BF	SCU_PAGE							WDTH
BE	BG	COCON						WDTL
BD	BCON	FEAH		MODSUSP				WDTWINB
BC	NMISR	FEAL						WDTREL
BB	NMICON	PASSWD		PMCON2				WDTCON
BA	EXICON1	CMCON		MODPISEL2				
B9	IPH							
B8	IP							
B7	EXICON0	PLL_CON		MODPISEL1				MD5_MR5
B6	IRCON2	OSC_CON						MD4_MR4
B5	IRCON1	PMCON1		IRCON4				MD3_MR3
B4	IRCON0	PMCON0		IRCON3				MD2_MR2
B3	MODPISEL	ID		XADDRH				MD1_MR1
B2	PORT_PAGE							MD0_MR0
B1	P3_DIR	P3_PUDEN	P3_ALTSEL1					MDUCON
B0	P3_DATA	P3_PUDSEL	P3_ALTSEL0	P3_OD				MDUSTAT
AF	SSC_BRH							
AE	SSC_BRL							
AD	SSC_RBL							
AC	SSC_TBL							
AB	SSC_CONH							
AA	SSC_CONL							
A9	SSC_PISEL							
A8	IEN0							
A7	CCU6_CMPMODIFH	CCU6_TCTR0H	CCU6_MCMCTR					
A6	CCU6_CMPMODIFL	CCU6_TCTR0L	CCU6_PSLR					
A5	CCU6_ISRH	CCU6_T12DTCH	CCU6_ISSH					
A4	CCU6_ISRL	CCU6_T12DTCL	CCU6_ISSL	CCU6_PISEL2				
A3	CCU6_PAGE							
A2	EO							
A1	P2_DIR	P2_PUDEN						CD_CON
A0	P2_DATA	P2_PUDSEL						CD_STATC
9F	CCU6_MCMOUTSH	CCU6_T13PRH	CCU6_INPH	CCU6_PISEL0H				CD_CORDZH
9E	CCU6_MCMOUTSL	CCU6_T13PRL	CCU6_INPL	CCU6_PISEL0L				CD_CORDZL
9D	CCU6_TCTR4H	CCU6_T12PRH	CCU6_IENH	CCU6_ISH				CD_CORDYH
9C	CCU6_TCTR4L	CCU6_T12PRL	CCU6_IENL	CCU6_ISL				CD_CORDYL
9B	CCU6_CC63SRH	CCU6_CC63RH	CCU6_T12MSELH	CCU6_MCMOUTH				CD_CORDXH
9A	CCU6_CC63SRL	CCU6_CC63RL	CCU6_T12MSELL	CCU6_MCMOUTL				CD_CORDXL
99	SBUF							
98	SCON							
97								
96								
95								
94								
93	P5_DIR	P5_PUDEN	P5_ALTSEL1					
92	P5_DATA	P5_PUDSEL	P5_ALTSEL0	P5_OD				
91	P1_DIR	P1_PUDEN	P1_ALTSEL1					
90	P1_DATA	P1_PUDSEL	P1_ALTSEL0	P1_OD				
8F	SYSCON0							
8E								
8D	TH1							
8C	TH0							
8B	TL1							
8A	TL0							
89	TMOD							
88	TCON							
87	PCON							
86	P0_DIR	P0_PUDEN	P0_ALTSEL1					
85								
84								
83	DPH							
82	DPL							
81	SP							
80	P0_DATA	P0_PUDSEL	P0_ALTSEL0	P0_OD				

Interrupt Vector Addresses				
Interrupt Mode	**Vector Address**	**Assignment for XC888**	**Enable Bit**	**SFR**
NMI	0073h	Watchdog Timer NMI	NMIWDT	NMICON
		PLL NMI	NMIPLL	
		FLASH NMI	NMIFLASH	
		VDDC Prewarning NMI	NMIVDD	
		VDDP Prewarning NMI	NMIVDDP	
		FLASH ECC NMI	NMIECC	
XINTR0	0003h	External interrupt 0	EX0	IEN0
XINTR1	000Bh	Timer 0	ET0	
XINTR2	0013h	External interrupt 1	EX1	
XINTR3	001Bh	Timer 1	ET1	
XINTR4	0023h	UART	ES	
XINTR5	002Bh	T2	ET2	
		UART Fractional devider (normal divider overflow)		
		MultiCAN Node 0		
		LIN		
XINTR6	0033h	MultiCAN NOdes 1 and 2	EADC	IEN1
		ADC[1:0]		
XINTR7	003Bh	SSC	ESSC	
XINTR8	0043h	External Interrupt 2	EX2	
		T21		
		CORDIC		
		UART 1		
		UART 1 Fractional Devider (normal divider overflow)		
		MDU [1:0]		
XINTR9	004Bh	External Interrupt 3	EXM	
		External Interrupt 4		
		External Interrupt 5		
		External Interrupt 6		
		MultiCAN Node 3		
XINTR10	0053h	CCU6 INP0	ECCIP0	
		MultiCAN Node 4		
XINTR11	005Bh	CCU6 INP1	ECCIP1	
		MultiCAN Node 5		
XINTR12	0063h	CCU6 INP2	ECCIP2	
		MultiCAN Node 6		
XINTR13	006Bh	CCU6 INP3	ECCIP3	
		MultiCAN Node 7		



Note: The pins shaded in blue are not available in the PG-TQFP-48 package.