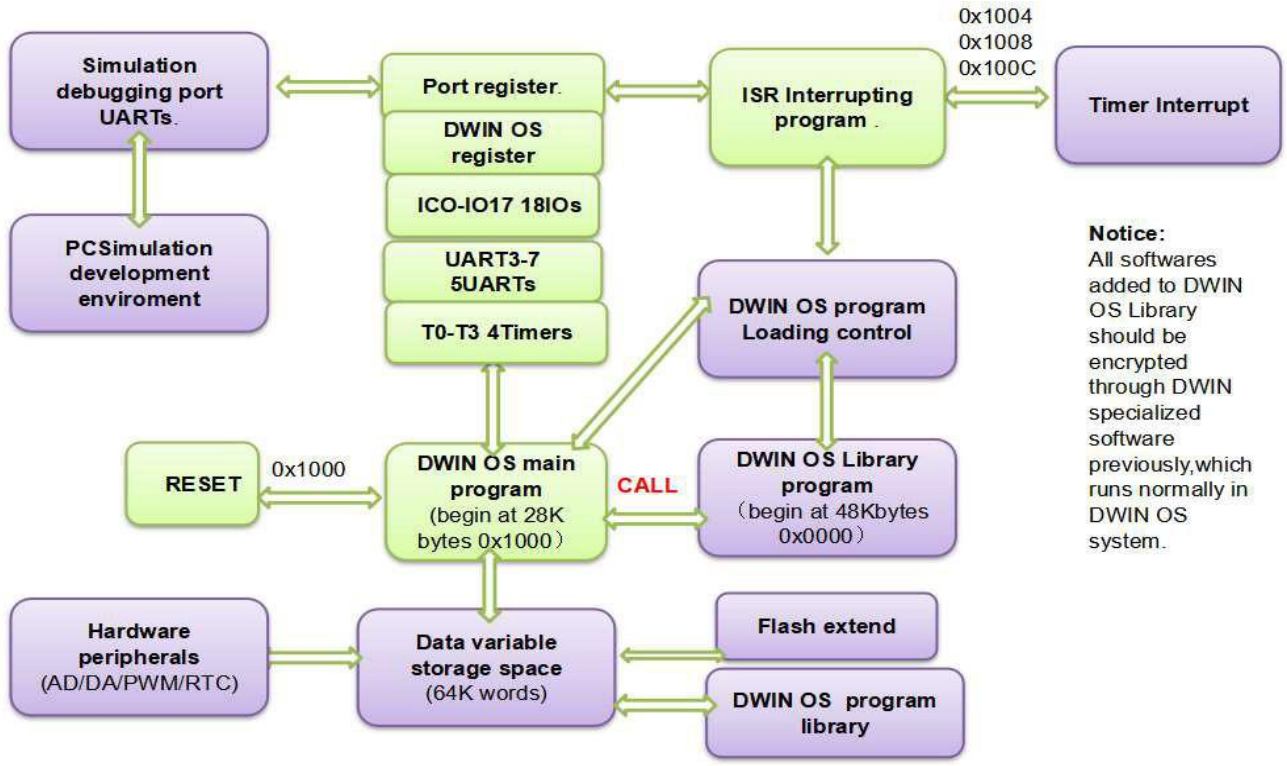
T5L DWIN OS development guide

2022/05/18

Contents

1 DWIN OS platform structure



(1) DWIN OS Cord Space Definition

Cord Address	Definition	Description
0x0000-0x0FFF	L2_Cache	Program automatic loads and calls space, 4 KB.
0x1000	RESET	The first address of program operating after reset, placing one GOTO command and jumping to the main program.
0x1004	T0 _INT	T0 INT program entrance address, applying GOTO command and jumping to T0 to interrupting service program.
0x1008	T1 _INT	T1 INT program entrance address, applying GOTO command and jumping to T1 to interrupting service program.
0x100C	T2 _INT	T2 INT program entrance address, applying GOTO command and jumping to T2 to interrupting service program.
0x1020-0x107F	Reserved	Reserve
0x1080-0x7FFF	Main Code	Main program cord space

(2) Subroutine Nested is transferred (including Interrupted program) up to level 127.

(3) Typical program Structure

```
        ORG     1000H
GOTO    MAIN
GOTO    T0INT;Discontinuity produce,jumping to T0 to interrupting program,must use


GOTO, can not use CALL NOP; T1 Discontinuity do not be used
GOTO    T2INT; Discontinuity produces, jumping to T0 to interrupting service program.
      ORG     1080H
MAIN: NOP             ;Main program
      start GOTO      MAIN
T0INT:  NOP           ;T0 interrupt handling
        program RETI          ;use RETI
        end,can not use RET.
T2OINT: NOP           ;T1 interrupt handling
        program RETI
```

If interrupt were not adopted(closing interrupting), code space of 0x1004-0x107F could be used at will.

If the main program needed to operating break point simulation, interrupt shall be closed. Otherwise, timer keeps running under simulation status, opening interrupt would cause the main program failed to operating break point simulation.

2. DWIN OS Debugging Port(UART2)

System Debug serial port UART2's mode is 8N1,baud rate can be installed, data frame is made up by 4 parts.

Explanation of UART2 Debug port instruction below.

Data	1	2	3	4	5
Definition	Frame Header	Data length	Command	Data	CRC check (optional)
Data Length	2	1	1	n	2
Description	Defined by R3 & RA in CONFIG.TXT	Data length, including command data and checksum	0x80-0x84		
For example	5A A5	04	83	00 10 04	25 A3

Instruction	Data	Description
0x80	Issued:ADR(0x00-0x08)+ADR(0x00-0xFF)+ Data_Pack	Data length
	Respond:0x4F 0xaB	Write Instruction respond
0x81	Issued:ADR (0x00-0x08)+ADR(0x00-0xFF)+ RD_LEN（0x01-0xFB)	Read data in designated addresses in register.
	Respond:ADR(0x00-0x08)+ADR(0x000xFF)+ RD_LEN+Data_Pack	Data respond
0x82	Issued:First ADR(0x0000-0xFFFF)+Data_Pack	Write data in designated addresses in variable
	Respond:0x4F 0xaB	Write Instruction respond
0x83	Issued:First ADR(0x0000-0xFFFF)+RD_LEN (0x01-0x7D)	Read data in designated addresses in variable SRAM.

Respond: First ADR+Variable data word length+Variable data read

Data respond Register page is defined as follow

Register page ID	Definition	Description
0x00-0x07	Data Register	256 per group,R0-R255
0x08	Port Register	DR0-DR255 Details in DWIN OS development guide basing on T5, 3.4 port register definition description.

.

3 Storage Space

3.1Users Data Library

Users data library includes two parts:

(1) Flash on T5L chips visits through system variation port, all DWIN OS based in T5 supporting.

(2) High-capacity data base or material storage locating at NAND Flash, which visits visits through system variation port, space room depends on hardware platform.

3.2 Data Variable Space

Data variable space is a biggest 128Kbytes double-port RAM. 2 CPU exchanges data by data variable space, definition list by district is as follow

At all DWIN OS platform basing on T5L CPU, the first 16-word definition of system variable port is unified, as follow.

Address	Definition	Length	Description
0x00	Device_ID	4	
0x04	System_Reset	2	0x55AA 5AA5=Reset T5L ;
0x06	OS_Update_CMD	2	D3:0x5A first updates DWIN OS program(inserting into ram Flash), clearing after CPU operation finish. D2:Fixing as 0x10, DWIN OS code should be start from 0x1000. D1:0:The initial data variation space address of store-and-update code should be even.
0x08	NOR_Flash_RW_ CMD	4	D7:operation mode 0x5A=read 0xA5=write, clearing after CPU operation finish. D6:4:The initial address of ram Nor Flash data base should be even. 0x000000-0x02:7FFE, 192KWords. D3:2:The initial data variation space address should be even. D1:0:The length of read and write word should be even.
0x0C	UART2_Set	2	D3=0x5A start UART2 Serial mode setting,just used for GUI CPU settingUART2 mode after Reset ,OS can not operate itself. D2=Serial mode,0x00=8N1. D1:D0=Baud rate value,Baud rate value=3225600/Baud rate setting.
0x0E	Reserve	1	
0x0F	Ver	1	Application software version.D1 refers to GUI soft version,D0 refers to DWIN OS software version.

3.3 Port Register

Basing on T5L, DWIN OS has a port register page, with 256 port registers, as quick visit port for hardware resources.

DR#	Length	R/W	Definition	Description
0	1	R/W	REG_Page_Sel	8 register pages of OS change, DR0=0x00-0x07
1	1	R/W	SYS_STATUS	System status register, defining as bit-wise: .7 CY carry flag. 6 DGUS screen variation automatic uploading control 1=close 0=open
2	14	- -	Reserve	Access forbidden.
16	1	R	UART3_TTL_S	Serial received frame overtime timer status:: 0x00= received frame overtime timer overflowing others=no overflowing It should be done first that applying RDXLEN command reads received length. When length is not 0, then checking overtime timer status.
17	1	R	UART3_TTL_S	
18	1	R	UART4_TTL_S	
19	1	R	UART5_TTL_S	
20	1	R	UART6_TTL_S	
20	1	R	UART7_TTL_S	
21	1	- -	Reserve	
22	1	R	UART3_TX_LEN	UART3 the applying depth(Bytes) of buffer sending buffer size is 256Bytes,users read only.
23	1	R	UART4_TX_LEN	UART4 the applying depth(Bytes) of buffer sending buffer size is 256Bytes,users read only.
24	1	R	UART5_TX_LEN	UART5 the applying depth(Bytes) of buffer sending buffer size is 256Bytes,users read only.
25	1	R	UART6_TX_LEN	UART6 the applying depth(Bytes) of buffer sending buffer size is 256Bytes,users read only.
26	1	R	UART7_TX_LEN	UART7 the applying depth(Bytes) of buffer sending buffer size is 256Bytes,users read only.
27	1	- -	Reserve	
28	1	R/W	UART3_TTL_SET	UART3 the time of received frame overtime timer. Unit 0.5mS,0x01-0xff, power-on set as 0x0A.
29	1	R/W	UART4_TTL_SET	UART4 the time of received frame overtime timer. Unit 0.5mS,0x01-0xff, power-on set as 0x0A.
30	1	R/W	UART5_TTL_SET	UART5 the time of received frame overtime timer. Unit 0.5mS,0x01-0xff, power-on set as 0x0A.
31	1	R/W	UART6_TTL_SET	UART6 the time of received frame overtime timer. Unit 0.5mS,0x01-0xff, power-on set as 0x0A.
32	1	R/W	UART7_TTL_SET	UART7 the time of received frame overtime timer. Unit 0.5mS,0x01-0xff, power-on set as 0x0A.
33	1	- -	Reserve	
34	1	R/W	T0	8bit user timer 0,++counting,datum 10uS
35	2	R/W	T1	16bit user timer 1,++ counting,datum 10uS
37	2	R/W	T2	16bit user timer 2,++ counting,datum designed by users through CONFIG command

39	2	R/W	T3	16bit user timer 3,++ counting,datum designed by users through CONFIG command
41	1	R/W	CNT0_Sel	relevant position 1 choosing related I/O to counting changes, corresponding IO7-IO0
42	1	R/W	CNT1_Sel	relevant position 1 choosing related I/O to counting changes, corresponding IO7-IO0
43	1	R/W	CNT2_Sel	relevant position 1 choosing related I/O to counting changes, corresponding IO15-IO8
44	1	R/W	CNT3_Sel	relevant position 1 choosing related I/O to counting changes, corresponding IO15-IO8
45	1	R/W	Int_Reg	Interrupting control register. .7= interrupt main switch 1= enable(open or not depending on single interrupting control position) 0=ban. .6=Timer INT0 Enable 1= interrupt timer 0 interrupt on 0= interrupt timer 0 interrupt off. .5=Timer INT1 Enable 1=interrupt timer 1 interrupt on 0=interrupt timer 1 interrupt off. .4=Timer INT2 Enable 1=interrupt timer 2 interrupt on 0=interrupt timer 2 interrupt off.
46	1	R/W	Timer INT0 Set	8bit timer interrupt 0 settings value, interrupt time=Timer_INT0_Set*10uS, 0x00=256.
47	1	R/W	Timer INT1 Set	8bit timer interrupt 1 setting value, interrupt time=Timer_INT1_Set*10uS, 0x00=256.
48	2	R/W	Timer INT2 Set	16bit timer interrupt 2 setting value,interrupt time=（Timer_INT2_Set+1）*10uS.
50	10	R/W	Polling_Out0_ Set	Firstly, IO0-IO15 scans output configuration timely, 10 bites each. D9（DR50）:0x5A=scan output applying, other as no applying. D8: Outputting register page of data, 0x00-0x07. D7: Outputting start and end address of data, 0x00-0 xFF D6: Outputting the word length of data, 0x01-0x80, each data 2 Bytes corresponding to IO15-IO0. D5-D4:IO15-IO0 output aisle choosing, needing output aisle, corresponding bit set as 1. D3-D2:single outputting interval T,unit as（T+1）*10uS. D1-D0:Outputting cycle counting designed, it minus 1 after finishing 1 cycle every time, then outputting 0 till minus to 0.
60	10	R/W	Polling_Out1_ Set	Second : IO0-IO15 scans output configuration timely.
70	9	- -	Reserve	
80	6	R/W	IO6 Trigger time	D5=0x5A refers to IO6 is caught once, down edge trigger. D4:D3= IO15-IO0's condition when triggered. D2:D0= the catching time of system timer 0x000000-0x00FFFF cycle,unit as 1/41.75uS.
86	6	R/W	IO6 Trigger time	D5=0x5A refers to IO7 is caught once, down edge trigger. D4:D3= IO15-IO0's condition when triggered. D2:D0= the catching time of system timer 0x000000-0x00FFFF cycle,unit as 1/41.75uS.
92	37	- -	Reserve	
129	3	R/W	IO_Status	The real-time status of IO19-IO0 's

132	2	R/W	CNT0	CNT0 changing counting value, resetting to 0x0000 when counting to 0xFFFF.
134	2	R/W	CNT1	CNT1 changing counting value, resetting to 0x0000 when counting to 0xFFFF
136	2	R/W	CNT2	CNT2 changing counting value, resetting to 0x0000 when counting to 0xFFFF
138	2	R/W	CNT3	CNT3 changing counting value, resetting to 0x0000 when counting to 0xFFFF
140	116	- -	Reserve	

4 DWIN OS Assembly Instruction Set

- R#means DWIN OS in present register page,any or any group of 256 register,R0-R255;
- DR#means one or any group of 256 port register, DR0-DR255;
- < >means Immediate number,in the Assembly code,100,0x64,64H,064H all means 10 hex data 100.
- Directives:ORG DB DW.
- Use; as remark.
- Description of Variable and data type can be visited by DWIN OS as following:

Variable Type	Mark	Type	Space	Description
DWIN OS register	R0-R255	Byte	2048 Bytes	Divided into 8 pages,Control page by DR0 port register
Port register	DR0-DR255	Byte	256 Bytes	
Data variable space	XRAM	Word	64K Words	Range of address:0x0000-0xFFFF
User data library	LIB	Word	Depend on hardware	

When T5 CPU runs at the speed of 200MHz, the average operating time of one DWIN OS command is about 125nS (8MIPS).

4.1 Data Exchange Command

Command	Code	Number	Description
Data exchange between Variables & Registers	MOVXR	R#,<MOD>,<NU M>	R#:Register or Register group. <MOD>:0=Register to variable1=Variable to register. <NUM>:exchange data word （ Word ） length,0x00-0x80; When<NUM>is 0x00 ,data length depends on R9. Data variable pointer is defined by R0:R1 register. MOVXR R20,0,2
load N 8bit Immediate number to register group	LDBR	R#,<DATA> ,<NUM>	R#:Register or Register group. <DATA>:Data need loading. <NUM>:Number of Register need loading, 0x00 means 256. LDBR R8,0x82,3
Load several 16 bit numbers to Registers	LDWR	R#,<DATA>	R#:Register group. <DATA>:Number of that loading LDWR R8,1000 LDWR R8,-300
Load address code space	LDADR	<Address>	Load <Address> to R5:R6:R7 LDADR TAB LDADR 0x123456
Look up in Program Space (Program Space to DWIN_OS Registers)	MOVC	R#,<NUM>	R#:Register or Register group. <NUM>:byte data length Lookup table return Address pointer is defined by R5:R6:R7 register MOVC R20,10 Attention, code after 0x1000 can not read code content before 0x1000.
Data transfer from Register to DGUS Register	MOV	R#S,R#T,<NUM>	R#S:Origin register or register group R#T:goal register or register group. <NUM>:Font data length exchanged,,0x00 means length is defined by R9 register. MOV R8,R20,3
Register to port register	MOVRD	R#,DR#,<NUM>	R#:Register or Register group; DR#:Port Register or Register group; <NUM>:Font data length exchanged,0x00 means length is defined by R9 register. MOVRD R10,3,2
port register to Register	MOVDR	DR#,R#,<NUM>	R#:register or register group; DR#:port register or register group; <NUM>:Font data length exchanged,0x00 means length is defined by R9 register. MOVDR 3,R10,2
Exchange data between data variable	MOVXX	<NUM>	<NUM>:exchange（Word）length of data. <NUM> 0 means length is defined by R8:R9 register Origin variable address length is defined by R0:R1 register. Goal variable address is defined by R2:R3. When the distance between source address and target address is shorter than the length of moving data, the later length shall not longer than 32. MOVXX 100
Registers Indexed Addressing	MOVA		R2 stipulate origin address of register（group） R3 stipulate rule goal address of register （ group ） R9 stipulate data length exchanged,bytes. MOVA

4.2 Computation Command

Command Function	Code	Number	Description
32bit integers addition	ADD	R#A,R#B,R#C	C=A+B, A,B are 32bit integers, C is 64bit integer. e.g.: ADD R10, R20, R30
32bit integers subtraction	SUP	R#A,R#B,R#C	C=A-B, A,B are 32bit integers, C is 64bit integer. e.g.: SUB R10, R20, R30
64bit MAC for long integers	MAC	R#A,R#B,R#C	C=(A*B+C), A, B are 32bit integers, C is 64bit integer. e.g.: MAC R10, R20, R30
64bit integers division	DIV	R#A,R#B,<MOD>	A/B, A is quotient, B is reminder. A and B are 64bit register. <MOD>: 0: The quotient will not be rounded. 1: The quotient WILL BE ROUNDED. e.g.: DIV R10, R20, 1
Expand Variable to 32bit	EXP	R#S,R#T,<MOD>	Expand the data in R#S to 32bit and save to R#T R#S: Source register(s) R#T: Target register <MOD>: Data type of R#S. 0=8Bit unsigned;1=8bit signed 2=16bit unsigned;3=16bit integer e.g.: EXP R10, R20, 2
32bit unsigned MAC	SMAC	R#A,R#B,R#C	C=A*B+C A and B are 16bit unsigned integer, C is 32bit unsigned integer. e.g.: SMAC R10, R20, R30
Register self-increase	INC	R#,<MOD>,<NUM>	R#=R#+NUM, unsigned self-increasing calculation <MOD>: Data type of R#; 0=8bit;1=16bit e.g.: INC R10, 1,5
Register self-decrease	DEC	R#,<MOD>,<NU M>	R#=R#-NUM, unsigned self-decreasing calculation <MOD>: Data type of R#; 0=8bit;1=16bit e.g.: DEC R10, 0,1
Square root count	SQRT	R#A,R#B	Count a 64 bit unsigned R#A 's Square root and reserve it into R#B. R#A:Reserve 8 Bytes unsigned; R#B:Reserve 4 Bytes unsigned result. e.g: SQRT R80,R90

4.3 Logic Operation Command

Command Function	Code	Number	Description
Logical Calculation: AND	AND	R#A,R#B,<NUM>	A=A AND B, Logical “AND” calculation for series of Registers. <NUM>: Data length of R#A, R#B in BYTES e.g.: AND R10, R20,1
Logical Calculation: OR	OR	R#A,R#B,<NUM>	A=A OR B, Logical “OR” calculation for series of Registers. <NUM>: Data length of R#A, R#B in BYTES e.g.: OR R10, R20,1
Logical Calculation: XOR	XOR	R#A,R#B,<NUM>	A=A XOR B, Logical “XOR” calculation for series of Registers. <NUM>: Data length of R#A, R#B in BYTES e.g.: XOR R10, R20,1
Left ring move	SHL	R#,<NUM>,<BIT_NUM>	Turn R#point<NUM> register left and ring move <BIT_NUM>bit. eg :SHL R10,2,1
Right ring move	SHR	R#,<NUM>,<BIT_NUM>	Turn R#point <NUM>register right and ring move<BIT_NUM>bit. eg :SHR R10,2,1

4.4 Data Processing Command

Command Function	Code	Number	Description
Sequence comparison	TESTS	R#A,R#B,<NUM>	Compare the values in R#A and R#B by sequence. If not match, return the current address of R#A to R0 register; If match, return 0x00 to R0 register. R#A: Starting register for register series A; R#B: Starting register for register series B; <NUM>: max length for data comparison. e.g.: TESTS R10, R20, 16
Integer Linear Equation	ROOTL		Calculate the Y value according to the given X value, which is a point on the line defined by (X0, Y0) and (X1, Y1) in 16bit integer. Input: X=R10, X0=R14, Y0=R16, X1=R18, Y1=R20 Output: Y=R12 e.g.: ROOTLE
ANSI CRC-16	CRCA	R#S,R#T,R#N	Perform ANSI CRC-16 calculation on series of Registers. ANSI CRC-16(X16+X15+X2+1) R#S: Registers for Input R#T: Registers to hold the result, 16bit, LSB mode. R#N: Save the length for CRC byte data, 8bit e.g.: CRCA R10, R80, R9
CCITT CRC-16	CRCC	R#S,R#T,R#N	Perform CCITT CRC-16 calculation on series of Registers. CCITT CRC-16(X16+X12+X5+1) R#S: Registers for Input R#T: Registers to hold the result, 16bit, MSB mode. R#N: Save the length for CRC byte data, 8bit e.g.: CRCC R10, R80, R9
HEX transfer ASCII String	HEXASC	R#S,R#T,<MOD>	R#S: 32bit Integer needed transferring; R#T: ASCII string register group transferred ; <MOD>:Transfer mode,high 4bit is length of Integer bit,low 4bit is number of Decimal. ASCII string transferred with symbol,Right alignment,empty is filled with 0x20. To data 0x12345678, <MOD>=0x62 transferred result is +054198.96 <MOD>=0xF2 transferred result is +3054198.96 HEXASC R20,R30,0x62
Convert ASCII string to HEX characters	ASCHEX	R#S,R#T,<LEN>	Convert ASCII String to signed 64 bit HEX data. R#S: Starting address for registers stored ASCII Strings R#T: A 64bits register to hold the output 64bit Hex data. <LEN>: The length for ASCII string, include sign bit and decimal point. 0x01-0x15. e.g.: ASCHEX R10, R80, 0x05
Algorithm deal	MATH	<MATH_ID>	transfer standard data-processing algorithm. <MATH_ID>:Algorithm type transferred （1）MATH_ID=0x00 forecast result R0:R1= VP pointer location that reserve Historical record data ,storage related to（x,y）order . Both are 16bit Integer. R2= data length of Historical record,0x02-0xFF. R3:R4=input X value,16bit Integer. R5:R6=output Y value （ forecast result ） . e.g:MATH 0,R20,R30

4.5 Process Controlling Command

Command Function	Code	Number	Description
None	NOP		None of operation NOP
Conditional bit jump	JB	R#,<Bit>,<TAB>	Evaluate the <bit> in R# register. If 1, jump to <NUM>; if 0, proceed to next instruction. R#: The register contains data to be evaluated. <Bit>: the index of the bit to be evaluated. 0x00-0x0F (MSB). <NUM>: Jump position control. No.7 control the direction: 1 = forward; 0=backward; result for NUM&0x7F indicates the number of instructions to jump. e.g.: JB R10, 15,TEST1 NOP TEST1: ADD R8,R12, R16
Variable conditional jump (Not Equal)	CJNE	R#A,R#B,<TAB>	Compare the value of 2 8bit registers (R#A and R#B). If equal, proceed to next instruction; if not equal, jump to <NUM>. e.g.: TEST1: NOP INC R10, 0, 1 CJNE R10,R11, TEST1
16bit Integer conditional jump (Less than)	JS	R#A,R#B,<TAB>	Compare the value for 2 bit integer in R#A and R#B. If A>=B, proceed to next instruction; If A<B, jump to <NUM> e.g.: JS R10, R12, TEST1 NOP TEST1: NOP
16bit comparison, <jumping	JU	R#A,R#B,<TAB>	Compare A、B 16bit the unsigned ,A>=B carry out next command,A<B jumping,range of jumping +/-127 commands. JU R10,R12,TEST1 NOP TEST1:NOP
Value conditional jump (Number and Variable)	IJNE	R#,<INST>,<TAB>	Compare the value in 8 bit Register and a instant Number <INST>. If equal, process to next instruction; if not equal, jump to <NUM>. e.g.: IJNE R10, 100, TEST1 NOP TEST1: NOP
Decrements> 0 jumping	DJNZ	R#,<NUM>,<TAB>	R#is 16bit ,Every count R#=R#-<NUM>,if R#> 0 jumping, In contrast,carry out next command,range of jumping +/-127 commands. TEST1:NOP DJNZ R10,1,TEST1
Return	RET		Return to main program by calling this function in sub-program. e.g.: RET
Interrupt program return	REIT		Interrupt program and return. RETI
Call sub-function	CALL	<PC>	Call sub-program in position of program counter(Maximum support 32 levels of Program Nesting) e.g.: CALL TEST
Direct Jump	GOTO	<PC>	<MOD>=0x00: Jump to <PCH:L> <MOD>=0x01: Relatively Jump to (PC+1+<PCH:L>) <MOD>=0x02: Relatively Jump to (PC+1-<PCH:L>) <PCH:L>=0xFFFF indicates the value is in R0:R1 e.g.: GOTO TEST1 NOP TEST1: NOP
Program end	END		DWIN OS program over command After carry out this command, PC pointer Reset to 0x1000 ,run again.same as software reset. END

Notice:

Interrupting program should apply GOTO, RETI command.

Subprogram calling must use CALL and RET command in Pairs, transferring program with GOTO and RET command will result in abnormal Stack overflow

4.6 Peripheral Operation Command

Command Function	Code	Number	Description
Serial setting	COMSET	<MODE/R#>,<BS>	Set serial port mode: <MODE>: high 4bit choose serial port needed to be set,3=UART3 … 5=UART5 low 4bit choice mode. 0x*0=N81,0x*1=E81,0x*2=O81,0x*3=N82 mode. <BS>: Baud rate setting value,2Bytes. Setting value= 6451200/ set Baud rate. To UART3,range of setting value=1-1023,lowest Baud rate 15.3kbps. To UART4-UART5,Setting value= 25804800/ set ,range of setting value 1-65535 Corresponding UART transceiver buffer will be cleared during each setting procedure. If<BS>=0x0000, then<MODE>will be register pointer, pointing 3 registers, in sequence corresponded to<MODE>,<BS> value. COMSET 0x30,136
Serial send	COMTXD	<COM>,R#S,R# N	Dispatching data to specified port. <COM>: choose port, 0-2 no support 3=UART3… 5=UART5 R#S: data register group to be sent. R#N:bytes register to be sent,8bit,register data 0x00 refers to sending 256 Bytes data. COMTXD 3,R10,R9
Check COM_Rx_FIFO	RDXLEN	<COM>,R#	Returning to COM, receiving buffer area (FIFO) and data bytes length (0-255) to R# register, 0x00 refers to no data. <COM>: choose port, 0-2 no support 3=UART3… 5=UART5 RDXLEN 3,R10
Read COM_Rx_FIFO	RDXDAT	<COM>,R#A,R#B	Receiving buffer (FIFO) from COM, then reading R#B bytes (01-255) to R#A register. <COM>: choose serial port, 0-2 no support,3=UART3… 5=UART5 RDXDAT 3,R11,R10

Direct serial transmit	COMTXI	<COM>,R#,<NUM>	<NUM> register content with R# points sends to COM. <COM>: choose serial port, 0-2 no support,3=UART3… 5=UART5 COMTXI 3,R20,16 <TYPE>: hardware type choice,just low 7bit effective. TYPE.7=0 refers to D1、D0 as immediate values. TYPE.7=1 refers to D1will be register pointer, pointing 2 registers, in sequence corresponded to <D1><D0> value.
Hardware setting	CONFIG	<TYPE>,<D1/R#>,<D0>	0x00: Setting I/O port mode. D1 chooses IO port,0x00-0x02 corresponds P0-P2, among them P0.7-P0.0 corresponds IO7-IO0 P1.7-P1.0 corresponds IO15-IO8 P2.1-P2.0 corresponds IO17-IO16 D0 is corresponded setting value. D0.X=1 as output (Push-Pull) D0.X=0 as input (Open Drain) 0x01: Setting timer. D1 chooses timer, and 0x02-0x03 corresponds T2, T3. D0 sets timer datum, unit as 1ms, 0x00 refers to 256. 0x02: LIB code loading. As variable space address, D1: D0 should be even. Loading to code space starting as 0x0000, adopting CALL 0x0000 to transfer. 0x03: DWIN OS code loading. As variable space address, D1: D0 should be even. Loading to code space starting as 0x0000. Loading code should be encrypted in advance by DWIN specialized tool, loading time as about 200uS. CONFIG 0,0,0x0F
IO operation:output	OUTPUT	<P#>,<MOD>,R# /<OUT>	Outputting to a specified IO port（8bit or 1bit）. <P#>: serial number of IO port. 0x00-0x02 corresponds P0-P2. <MOD>: outputting mode 0x00=8 bit output, output as immediate number <OUT>. 0x01=8 bit output, output is R# specified value. 0x*2=output R#.0, then put R# right ring shift once, <MOD> high 4bit as IO position. 0x*3=output R#.7, then put R# left ring shift once, <MOD> high 4bit as IO position. OUTPUT 0,0,0x55 ; P0（IO7-IO0）port outputs 01010101 OUTPUT 1,0x32,R2; R2.0 outputs to IO11, and R2 right ring shift once. Read content of specified IO port（8bit or 1bit) to register. <P#>: serial number of IO port, 0x00-0x02 corresponds P0-P2. <MOD>: Inputting mode 0x00=8 bit parallel inputting. 0x*2= R# right ring move once, reading R#.7, <MOD> high 4bit as IO position. 0x*3= R# left ring move once, read R#.0, <MOD> high 4bit as IO position. R#: register ID that IO port data read.
IO operation:input	INPUT	<P#>,<MOD>,R#	

5 Appendix Revision Record

Date	Content revision
2017.04.22	Publishing at the first time.
2017.04.24	Complementing description of common part of system variable port 0x000-0x000F
2017.06.26	Adding GUI CPU setting to UART2 Baud rate. Adding system status register and definition of Carry mark CY,which helps extend of next mathematics and Logical bit operations. COMSET,CONFIG command add register operation
2017.07.27	Adding timer T0-T2 interrupting. Adding function of input count and timing scan output of IO .
2017.09.02	Adding support of DGUS II UART2 touch variable Automatic upload.
2017.10.21	Adding IO6,IO7 Lower edge time catching function
2018.01.06	Correcting BUG of UART4-UART7 and Full duplex work complicatedly
2018.01.30	Adding DGUS screen Automatic uploading function Control bit（DR1.6）
2021.11.25	English version
2022.05.18	Format revision

If there is any question when using this file or DWIN product, or willing to know more about DWIN product news, feel free to contact us:

Hotline:86-4000189008

Website:www.dwin-global.com