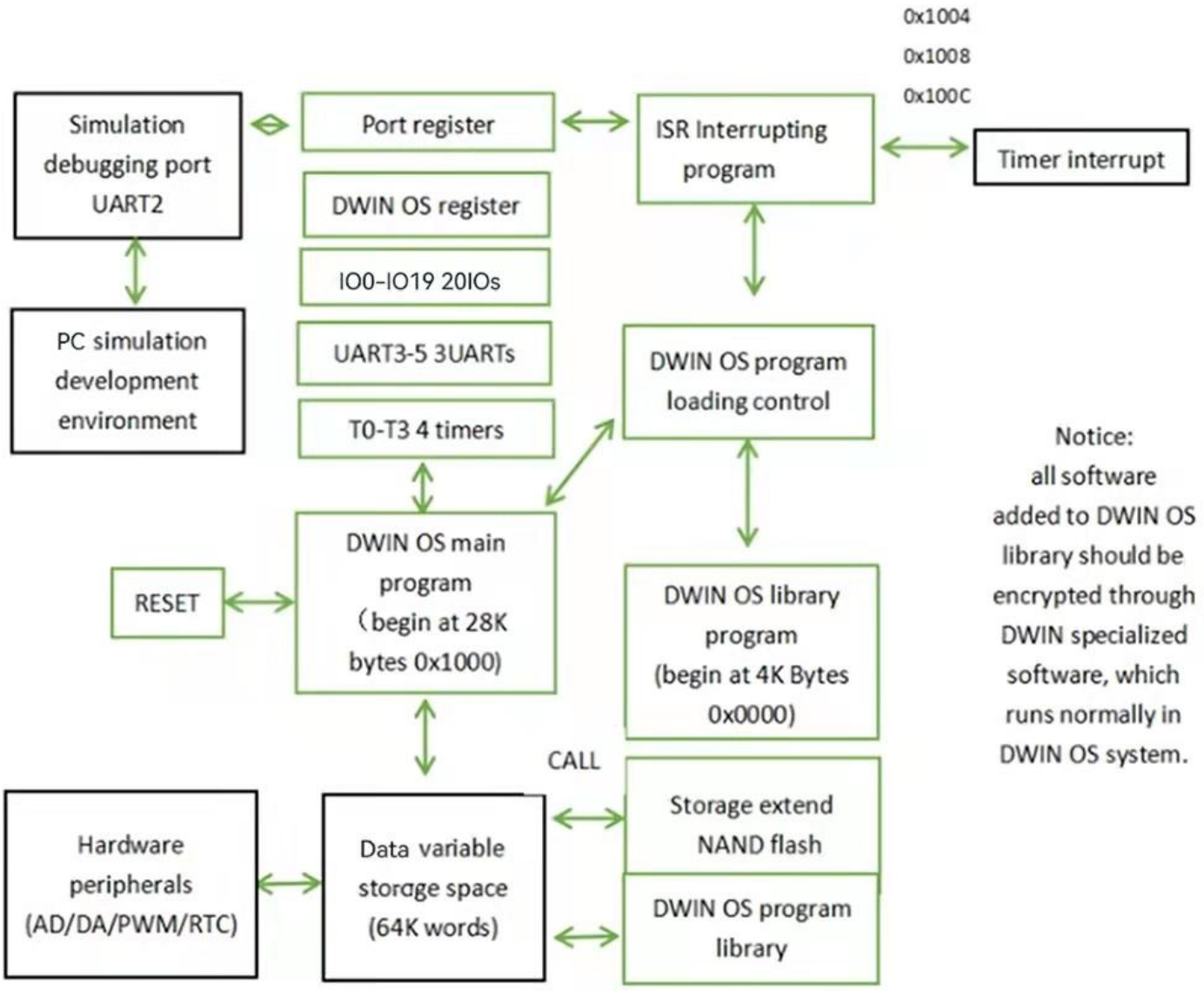
Development Guide of DWIN OS Platform Based on T5L CPU

Contents

1. DWIN OS Platform Structure



(1) DWIN OS Code Space Definition

Code Address	Definition	Description
0x0000-0x0FFF	L2_Cache	The space of program dynamically loads and calls, 4KB.
0x1000	RESET	The start address of program operating when reset, place one GOTO command to jump to the main program.
0x1004	T0 _INT	The entrance address of T0 INT program, applying GOTO command to jump to T0 interrupting service program.
0x1008	T1 _INT	The entrance addressT1 INT program, applying GOTO command to jump to T1 interrupting service program.
0x100C	T2 _INT	The entrance address T2 INT program, applying GOTO command to jump to T2 interrupting service program.
0x1020-0x107F	Reserved	Reserved.
0x1080-0x7FFF	Main Code	Main program code space.

(2) The max nested call of subroutine including interrupted program, up to grade 127.

(3) Typical program structure.

```
        ORG     1000H
GOTO   MAIN                 ; The first command must GOTO.
GOTO   T0INT                ; When interrupt, jumping to T0 to interrupting program, must use GOTO, can not use CALL.
NOP                         ; T1 Discontinuity not be used.
GOTO   T2INT                ; Discontinuity produces, jumping to T2 to interrupting service program.

    ORG     1080H
MAIN:   NOP                 ; Main program
        GOTO    MAIN

T0INT:  NOP                 ; T0 interrupt handling
        RET1                ; Use RETI end, can not use RET.

T1INT:  NOP                 ; T1 interrupt handling
        RET1
```

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

If the main program is needed to operate breakpoint simulation, the interrupt shall be closed. otherwise, the timer keeps running under simulation status, opening interrupt would cause the main program to fail operating breakpoint 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 5 parts.

Data	1	2	3	4	5
Definition	**Frame Header**	**Data Length**	**Command**	**Data**	**CRC Check (Optional)**
Data Length	2	1	1	n	2
Description	0x5AA5	Including command, data and check	0x80/0x81/0x82//0x83		
Example (without check)	5A A5	04	83	00 10 04	
Example (with check)	5A A5	06	83	00 10 04	25 A3

Explanation of UART2 debug port instruction below:

Instruction	Data	Description
0x80	Issue: ADR(0x00-0x08)+ADR(0x00-0xFF)+Data_Pack	Write data in designated addresses in register.
	Respond: 0x4F 0x4B	Write command respond.
0x81	Issue: ADR (0x00-0x08)+ADR(0x00-0xFF)+ RD_LEN（0x01-0xFB)	Read data in designated addresses in register.
	Respond: ADR(0x00-0x08)+ADR(0x00-0xFF)+RD_LEN+Data_Pack	Response with data.
0x82	Issue: First ADR(0x0000-0xFFFF)+Data_Pack	Write data in designated addresses in variable SRAM.
	Respond: 0x4F 0x4B	Write Instruction respond.
0x83	Issued: First ADR (0x0000-0xFFFF) + RD_LEN (0x01-0x7D)	Read the specified length of word data from the specified address in the variable space.
	Respond: First ADR + variable data word length + read variable data	Response with data.

Data respond register page is defined as follow:

Register Page ID	Definition	Description
0x00-0x07	Data Register	Each group 256, R0-R255
0x08	Port Register	DR0-DR255 Details in DWIN OS development guide basing on T5L, 3.4 port register definition description.

3.Storage Space

3.1Users Data Library

Include two parts:

(1) The flash in the T5L chip can be accessed by system variable interface and all DWIN OS based on T5 can be supported.

(2) A large database or data store located on an off-chip flash, accessed through a system variable interface, depending on the hardware platform.

3.2 Data Variable Space

The data variable space is a 128kbytes double-port RAM, the separated of T5L core can exchange data, the definition as below:

Variable Port Interval	Interval （K words）	Definition	Description
0x0000-0x03FF	1.0	System variable port	Hardware, storage visit control, data exchange. detailed definition is related to hardware platform.
0x0400-0x07FF	1.0	Reserved	Users do not apply.
0x0800-0x0BFF	1.0	Reserved	Users do not apply.
0x0C00-0x0FFF	1.0	Reserved	Users do not apply.
0x1000-0xFFFF	60	Users variable data/space	Users variable, storage read, write buffer and so on, users program by themselves.

At all DWIN OS platform based on T5L CPU, the first 16-word definition of system variable port is unified, shown as follows:

Address	Definition	Length	Description
0x00	Reserve	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 0x1000-0x0C7E should be even.
0x08	NOR_ Flash_RW_CMD	4	D7: operation mode 0x5A=read 0xA5=write, clearing after CPU operation finished. D6: 4: the initial address of ram Nor Flash data base should be even. 0x000000-0x02: FFFE, 256KWords. 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 means UART2 serial mode setting, only used for GUI CPU setting, UART2 mode after reset, OS can not operate itself. D2=serial mode, 0x00=8N1. D1: D0=baud rate value, baud rate value=3225600/set baud rate.
0x0E	Reserve	1	
0x0F	Ver	1	Application software version. D1 refers to CPU0 soft version, D0 refers to CPU1 software version.

Since variable memory is shared by two CPU cores, too frequent continuous reading and writing variable memory will seriously affect the processing efficiency of the CPU. Therefore, a 1mS timer interrupt is used to periodically query the data update of variable memory.

3.3 Register

DWIN OS based on T5L has a total of 2048 registers, divided into 8 pages for access, each page has 256 registers, corresponding to R0-R255.

3.4 Port Register

Based on T5L, DWIN OS has a port register page, with 256 port registers, as a 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, bitwise definition: .7 CY carry flag. .6 DGUS screen variation automatic uploading control 1=close 0=open
2	14	- -	System Reserve	Access forbidden.
16	1	R	UART3_TTL_Status	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_Status	
18	1	R	UART4_TTL_Status	
19	1	R	UART5_TTL_Status	
20	1	R	UART6_TTL_Status	
20	1	R	UART7_TTL_Status	
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	The 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, 0x00-0xFF 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	The second IO0-IO15 scans output configuration timely.
70	9	- -	Reserve	
80	6	R/W	IO6 Trigger time	D5=0x5A means that an IO 6 falling edge trigger is captured. 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	IO7 Trigger time	D5=0x5A means that an IO7 falling edge trigger is captured. 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

(1) R# means DWIN OS in present register page, any or any group of 256 register, R0-R255;

(2) DR# means one or any group of 256 port register, DR0-DR255;

(3) < >means Immediate number, in the Assembly code,100, 0x64, 64H, 064H all means 10 hex data 100.

(4) Pseudo directives: ORG DB DW.

(5) Use; as a comment symbol.

(6) 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	

(7)When T5L CPU runs at the speed of 200MHz, the average operating time of one DWIN OS command is about 125nS (8MIPS).

4.1Data Exchange

Command Function	Code	Number	Description
Data exchange between Variables & Registers	MOVXR	R#, <MOD>, <NUM>	R#:Register or Register group. <MOD>: 0=Register to variable 1=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 1 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 Look up 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	Without or 0x00	Register data exchange, according to register paging access: R2 stipulate origin address of register（group） R3 stipulate rule goal address of register group R9 stipulate data length exchanged, bytes. MOVA or MOVA 0x00
		0x01	Register data exchange, all registers as a 2KB data area to access: R0: R1stipulate origin address of register, 0x0000-0x7FF; R2: R3stipulate rule goal address of register, 0x0000-0x7FF; R9 stipulate date length exchanged, bytes, 0x00-0xFF, 0x00 show 256. Address high byte = source or destination register DR0 low byte = source or destination register address. MOVA 0x01
Register stack 256 bytes	PUSH	R#<NUM>	Save the <NUM> register data starting with R# to the data stack. PUSH R8, 4
	POP	R#<NUM>	Fetch data from the data stack to the <NUM> registers starting with R#. POP R8, 4

4.2 Computation

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	SUB	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,<NUM>0x00-0xFF. <MOD>: Data type of R#; 0=8bit 1=16bit E.g. INC R10, 1, 5

Register self-decrease	DEC	R#, <MOD>, <NUM>	R#=R#-NUM, unsigned self-decreasing calculation,<NUM>0x00-0xFF. <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
Floating-point and fixed-point integer conversion	FINT	R#F, R#I, <MOD>	Implement 1 floating point number and 1 64bit fixed point integer conversion. R#F: The r egister that holds the floating-point number, 32bit or 64bit; R#I : Register to store fixed-point integer, 64bit; <MOD>: .7 Represents the floating-point number format: 0=32bit single precision 1=64bit double precision. .6 Conversion type 0=Convert floating-point number to fixed-point integer 1=Convert fixed-point integer to floating-point number. .5 Undefined, write 0 .4-.0 The number of decimal places for fixed-point integers, 0x00-0x1F, up to 31 decimal places.

4.3 Logic Operation

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. E.g. SHL R10, 2, 1
Right ring move	SHR	R#, <NUM>, <BIT_NUM>	Turn R#point <NUM>register right and ring move<BIT_NUM>bit. E.g. SHR R10, 2, 1

4.4 Data Processing

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	ROOTLE		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. ROOTE
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 transfering; R#T: ASCII string register group transfered; <MOD>: Transfer mode,high 4bit is length of Integer bit, low 4bit is number of Decimal. ASCII string transfered with symbol, Right alignment, empty is filled with 0x20. To data 0x12345678, <MOD>=0x62 transfered result is+054198.96 <MOD>=0xF2 transfered 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 64-bit Hex data. <LEN>: The length for ASCII string, include sign bit and decimal point. 0x01-0x15. E.g. ASCHEX R10, R80, 0x05
Data processing	MATH	<MOD>, R#P, R#N	Process the data in the data storage area, and the data is a 16-bit unsigned number. R#P: 7 register data D0-D7 starting from R#P. D0The start address of the register where the processing result is saved. D1: D2 The next (future) data is at the first address (relative address, data location) of the data buffer the historical data will be read forward from the current position during processing). D3: D4 The start address of the data buffer area in the data storage area. D5: D6 The word length of a D6 data buffer area must not be less than the value of R#N. R#N: The number of data processing points (1-N words per point), 0x00-0xFF, 0x00 means 256. <MOD>=0x00 Calculate the average value, and the result is a 32-bit unsigned number with a unit of 1/65536. <MOD>=0x01 Calculate the max value, and the result is a 16bit unsigned number. <MOD>=0x02 Calculate the min value, and the result is a 16-bit unsigned number. <MOD>=0x03 Calculate the root mean square (RMS), and the result is a 32-bit unsigned number with a unit of 1/65536. <MOD>=0x04 According to y=k*x+b, carry out least square method parameter estimation. The data storage format is (x0, y0)...(xn, yn), and the return format is (k, b). k, b The returned result is a 32-bit integer, and the unit is 1/65536. <MOD>=0x05 Not support. <MOD>=0x06 Calculate the standard deviation (RMSE), and the result is a 32-bit unsigned number with a unit of 1/65536. MATH 0, R0, R10

4.5 Process Controlling

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, jump range +/-127 instructions. R#: the register contains data to be evaluated. <Bit>: the index of the bit to be evaluated. 0x00-0x0F (MSB). <TAB>: jump position. 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 8-bit registers (R#A and R#B). If equal, proceed to the 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 the 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
Decrement> 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 +/-127commands. TEST1: NOP DJNZ R10, 1, TEST1
Return	RET		Return to the main program by calling this function in the sub-program. E.g.RET
Interrupt program return	REIT		Interrupting program and return. RETI
Call sub-function	CALL	<PC>	Call sub-program in a position of program counter max support 32 levels of program nesting. E.g. CALL TEST
Direct jump	GOTO	<PC>	Program jump. If <PC>=0xFFFF, it means taking the position of <R5:R6:R7> as the reference, and R1: R0 as the relative PC. Pointer to jump. E.g. GOTO TEST1 NOP TEST1: NOP
Program end	END		DWIN OS program over command after carrying 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 commands in pairs, transferring the program with GOTO and RET command will result in an abnormal stack overflow.

4.6 Peripheral Operation

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. For UART3, Setting value= 6451200/ set baud, rate. setting value range = 1-1023. To UART4-UART5, setting value= 25804800/ set, range of setting value 1-65535. Corresponding UART's 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 sent to COM. <COM>: choose serial port, 0-2 no support, 3=UART3… 5=UART5 COMTXI 3, R20,16

Hardware setting	CONFIG	<TYPE>, <D1/R#>, <D0>	<TYPE>: hardware type choice, just low 7bit effective. TYPE.7=0 refers to D1, D0 as immediate values. TYPE.7=1 refers to D1 will be register pointer, pointing 2 registers. TYPE.6-TYPE.0 select the hardware type, 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 corresponds 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. 0x04: Unencrypted 512-byte OS code loading (program call or process management); D1: 00 is the variable space word address of the stored code, and D0: 00 is the code space byte address. Loading code should be encrypted in advance by DWIN specialized tool, loading time is about 200uS. CONFIG 0, 0, 0x0F
IO operation: output	OUTPUT	<P#>, <MOD>, R#/<OUT>	Outputting to a specified IO port（8-bit or 1bit）. <P#>: A serial number of IO port. 0x00-0x02 corresponds to P1-P3. <MOD>: outputting mode 0x00=8 bit output, output is 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.
IO operation: input	INPUT	<P#>, <MOD>, R#	Read the content of specified IO port（8-bit or 1bit) to register. <P#>: The serial number of IO port, 0x00-0x02 corresponds P1-P3. <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 read from IO port data. INPUT 1, 0, R20; IO15-IO8 outputs R20 register value INPUT 1, 0x32, R2; R2 right ring move once, reading R2.7 as IO11 status.

5. Appendix

Date	Content Revision	Software version
2019.02.12	First Issuance	V1.0

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

Hotline: 86-4000189008

Mail: dwinhmi@dwin.com.cn