Table of Contents

1 C Language Library

1.1 CONST defined in 7188.H

```
#define     NoError             0
#define     InitPinIsOpen       0
#define     InitPinIsNotOpen    1
#define     QueueIsEmpty        0
#define     QueueIsNotEmpty     1
#define     PortError          -1
#define     DataError          -2
#define     ParityError        -3
#define     StopError          -4
#define     TimeOut            -5
#define     QueueEmpty         -6
#define     QueueOverflow      -7
#define     PosError           -8
#define     AddrError          -9
#define     BlockError        -10
#define     WriteError        -11
#define     SegmentError      -12
#define     BaudRateError     -13
#define     CheckSumError     -14
#define     ChannelError      -15
#define     TimeIsUp            1
```

1.2 Functions By type

1.2.1 Functions for COM ports

Install comport driver

Function name	description
InstallCom	For any one of 4 comports
InstallCom1	For com1
InstallCom2	For com2
InstallCom3	For com3
InstallCom4	For com4

Uninstall comport driver

Function name	description
RestoreCom	For any one of 4 comports
RestoreCom1	For com1
RestoreCom2	For com2
RestoreCom3	For com3
RestoreCom4	For com4

Check if has data in comport input buffer

Function name	description
IsCom	For any one of 4 comports
IsCom1	For com1
IsCom2	For com2
IsCom3	For com3
IsCom4	For com4

Read one byte data from comport input buffer

Function name	description
ReadCom	For any one of 4 comports
ReadCom1	For com1
ReadCom2	For com2

Function name	description
ReadCom3	For com3
ReadCom4	For com4

Send one byte data to comport

Function name	description
ToCom	For any one of 4 comports
ToCom1	For com1
ToCom2	For com2
ToCom3	For com3
ToCom4	For com4

Send n bytes data to comport

Function name	description
ToComBufn	For any one of 4 comports
ToCom1Bufn	For com1
ToCom2Bufn	For com2
ToCom3Bufn	For com3
ToCom4Bufn	For com4

Send string to comport

Function name	description
ToComStr	For any one of 4 comports
ToCom1Str	For com1
ToCom2Str	For com2
ToCom3Str	For com3
ToCom4Str	For com4

Clear all data in comport input buffer

Function name	description
ClearCom	For any one of 4 comports
ClearCom1	For com1
ClearCom2	For com2
ClearCom3	For com3
ClearCom4	For com4

Wait data send out to comport transmit over

Function name	description
WaitTransmitOver	For any one of 4 comports
WaitTransmitOver1	For com1
WaitTransmitOver2	For com2
WaitTransmitOver3	For com3
WaitTransmitOver4	For com4

Set the direction of RS-485 bus

Function name	description
Set485DirToTransmit	Set to transmit mode, For com1 & com2
Set485DirToReceive	Set to receive mode, For com1 & com2

Check if comport output buffer is empty

Function name	description
IsCom3OutBufEmpty	For com3
IsCom4OutBufEmpty	For com4

Get the data size in comport input buffer

Function name	description
DataSizeInCom	For any one of 4 comports
DataSizeInCom1	For com1
DataSizeInCom2	For com2
DataSizeInCom3	For com3
DataSizeInCom4	For com4

Communication with 7000's module

Function name	description
SendCmdTo7000	Send command to 7000's module
ReceiveResponseFrom7000	Receive response from 7000's module

1.2.2 Functions for LED

For red LED ON/OFF

Function name	description
LedOff	Set LED Off
LedOn	Set LED On

1.2.3 Functions for 5-Digit LED

Function name	description
Init5DigitLed	Initialize 5-Digit Led
Show5DigitLed	Show digit number(0-F)
Show5DigitLedWithDot	Show digit number and the dot '.'
Show5DigitLedSeg	Show any segment
Set5DigitLedTestMode	Set to test mode, all segment will be turn ON
Set5DigitLedIntensity	Set the intensity of 5-Digit LED
Disable5DigitLed	Disable 5-Digit LED, all segment will be OFF
Enable5DigitLed	Back to normal mode

1.2.4 Functions for data storage

For NVRAM

Function name	description
ReadNVRAM	Read one byte data from NVRAM
WriteNVRAM	Write one byte data to NVRAM

For EEPROM

Function name	description
WriteEEP	Write one byte data to EEPROM
ReadEEP	Read one byte data from EEPROM

Function name	description
EnableEEP	Enable EEPROM, so it can be write
ProtectEEP	Set EEPROM to write protect

For flash memory

Function name	description
FlashReadId	Read the Flash Memory type
FlashWrite	Write one byte data to Flash Memory
FlashErase	Erase one sector of flash memory
FlashRead	Read one byte data from Flash Memory

1.2.5 Functions for WatchDog Timer

Function name	description
EnableWDT	Enable WatchDog timer
RefreshWDT	Refresh WatchDog timer
DisableWDT	Disable WatchDog timer
IsResetByWatchDogTimer	Check if 7188 is reset by WatchDog timer
IsResetByPowerOff	Check if 7188 is reset by Power Off

1.2.6 Functions for timer

Function name	description
TimerOpen	Install timer driver
TimerClose	Uninstall timer driver
TimerResetValue	Reset timer ticks to 0
TimerReadValue	Read timer ticks
StopWatchReset	Reset stopwatch timer value to 0
StopWatchStart	Start stopwatch
StopWatchStop	Stop stopwatch
StopWatchPause	Pause stopwatch

Function name	description
StopWatchContinue	Continue stopwatch
StopWatchReadValue	Read the timer value of stopwatch
CountDownTimerStart	Start countdown timer
CountDownTimerReadValue	Read the timer value of countdown timer
InstallUserTimer	Install user's timer function
InstallUserTimer1C	Install user's timer function on INT 0x1C
DelayTimeMs	Delay time, unit is 1ms(use software)
DelayMs	Delay time, unit is 1ms(use hardware)
Delay	Delay time, unit is 1ms(use hardware)
Delay_1	Delay time, unit is 0.1ms(use hardware)

1.2.7 Functions for stdio

Function name	description
getch4	For instead of _getch
kbhit4	For instaed of kbhit
ungetch4	For instead of ungetch
putch4	For instead of putchar

1.2.8 MISC functions

Function name	description
InitLib	Initial 7188 library
GetLibVersion	Get version number of 7188 library
Is7188	Check if the program is run on 7188
ReadInitPin	Read the INIT* pin status
hex_to_ascii[16]	Global variable,for convert 0-15 to '0'-'F'
ascii_to_hex	For convert '0'-'F' to 0-15
_MK_FP	Make a far pointer

2 7188(D)/DOS library reference

1.3 InstallCom

- **Description:**
  Install the interrupt driven COM driver. This driver will handle all hardware details and provide 1K QUEUE input buffers for every COM port. For COM3/4 also provide 256 bytes output buffer.
- **Declaration:**
  int InstallCom(**int** port, **unsigned long** baud, **int** data, **int** parity, **int** stop);
- **Input Parameter:**

Argument	What it does
Port	1/2/3/4 for COM 1/2/3/4
Baud	Baudrate, 1200,2400,…. 57600,115200
Data	Data bits. 6 or 7 or 8(6 only for COM1/2)
Parity	Parity bit. 0=None, 1=EVEN, 2=ODD 3=MARK,4=SPACE(3,4 only for COM1/2)
Stop	Stop bit. 1= 1 stop-bit, 2=2 stop-bit(2 only for COM1/2)

- **Return Value:**
  When install success return NoError.
  Others: return Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  demo2.c

1.4 InstallCom1

- **Description:**
  Install the interrupt driven driver for COM1. This driver will handle all hardware details and provide 1K bytes input buffer. But no output

buffer.

- **Declaration:**
  int InstallCom1(**unsigned long** baud, **int** data, **int** parity, **int** stop);
- **Input Parameter:**

Argument	What it does
Baud	Baudrate, 1200,2400,…. 57600,115200
Data	Data bits. 6 or 7 or 8
Parity	Parity bit. 0=None, 1=EVEN, 2=ODD, 3=MARK, 4=SPACE
Stop	Stop bit. 1= 1 stop-bit, 2=2 stop-bit

- **Return Value:**
  When install success return NoError.
  Others: return Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  None.

1.5 InstallCom2

- **Description:**
  Install the interrupt driven driver for COM2. This driver will handle all hardware details and provide 1K bytes input buffer. But no output buffer.
- **Declaration:**
  int InstallCom2(**unsigned long** baud, **int** data, **int** parity, **int** stop);
- **Input Parameter:**

Argument	What it does
Baud	Baudrate, 1200,2400,…. 57600,115200
Data	Data bits. 6 or 7 or 8
Parity	Parity bit. 0=None, 1=EVEN, 2=ODD, 3=MARK, 4=SPACE
Stop	Stop bit. 1= 1 stop-bit, 2=2 stop-bit

- **Return Value:**
  When install success return NoError.
  Others: return Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**

None.

1.6 InstallCom3

- **Description:**
  Install the interrupt driven driver for COM3. This driver will handle all hardware details and provide 1K bytes input buffer and 256 bytes output buffer. **The stop bit always is 1.**
- **Declaration:**
  int InstallCom3(**unsigned long** baud, **int** data, **int** parity);
- **Input Parameter:**

Argument	What it does
Baud	Baudrate, 1200,2400,…. 57600,115200
Data	Data bits. 7 or 8
Parity	Parity bit. 0=None, 1=EVEN, 2=ODD

- **Return Value:**
  When install success return NoError.
  Others: return Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  None.

1.7 InstallCom4

- **Description:**
  Install the interrupt driven driver for COM4. This driver will handle all hardware details and provide 1K bytes input buffer and 256 bytes output buffer. **The stop bit always is 1.**
- **Declaration:**
  int InstallCom4(**unsigned long** baud, **int** data, **int** parity);
- **Input Parameter:**

Argument	descriptions
Baud	Baudrate, 1200,2400,…. 57600,115200
Data	Data bits. 7 or 8

Argument	descriptions
Parity	Parity bit. 0=None, 1=EVEN, 2=ODD

- **Return Value:**
  When install success return NoError.
  Others: return Error code(negative value, Refer to Sec. 1.1)
- Demo program:
  None.

1.8 RestoreCom

- **Description:**
  Uninstall the COM port driver, and restore the old driver.
- **Declaration:**
  int RestoreCom(**int** port);
- **Input Parameter:**

Argument	descriptions
Port	1/2/3/4 for COM 1/2/3/4

- **Return Value:**
  When restore success return NoError.
  Others: return Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  demo2.c

1.9 RestoreCom1

- **Description:**
  Uninstall the COM1 port driver, and restore the old driver.
- **Declaration:**
  int RestoreCom1(**void**);
- **Input Parameter:**
  None.
- **Return Value:**
  When restore success return NoError.

Others: return Error code(negative value, Refer to Sec. 1.1)

- **Demo program:**
  demo2.c

1.10 RestoreCom2

- **Description:**
  Uninstall the COM2 port driver, and restore the old driver.
- **Declaration:**
  int RestoreCom2(**void**);
- **Input Parameter:**
  None.
- **Return Value:**
  When restore success return NoError.
  Others: return Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  demo2.c

1.11 RestoreCom3

- **Description:**
  Uninstall the COM3 port driver, and restore the old driver.
- **Declaration:**
  int RestoreCom3(**void**);
- **Input Parameter:**
  None.
- **Return Value:**
  When restore success return NoError.
  Others: return Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  demo2.c

1.12 RestoreCom4

- **Description:**
  Uninstall the COM4 port driver, and restore the old driver.
- **Declaration:**
  int RestoreCom4(**void**);
- **Input Parameter:**
  None.
- **Return Value:**
  When restore success return NoError.
  Others: return Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  demo2.c

1.13 IsCom

- **Description:**
  Check is there any data in the comport input queue buffer .
- **Declaration:**
  int IsCom(**int** port);
- **Input Parameter:**

Argument	Descriptions
port	1/2/3/4 for COM 1/2/3/4

- **Return Value:**
  QueueIsEmpty ..... QUEUE is empty
  QueueIsNotEmpty ..... QUEUE is not empty
  Others..... Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  demo2.c

1.14 IsCom1

- **Description:**
  Check is there any data in the com1 input queue buffer .
- **Declaration:**
  int IsCom1(**void**);
- **Input Parameter:**
  None.
- **Return Value:**
  QueueIsEmpty .....QUEUE is empty
  QueueIsNotEmpty .....QUEUE is not empty
  Others .....Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  None.

1.15 IsCom2

- **Description:**
  Check is there any data in the com2 input queue buffer .
- **Declaration:**
  int IsCom2(**void**);
- **Input Parameter:**
  None.
- **Return Value:**
  QueueIsEmpty .....QUEUE is empty
  QueueIsNotEmpty .....QUEUE is not empty
  Others .....Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  None.

1.16 IsCom3

- **Description:**
  Check is there any data in the com3 input queue buffer .
- **Declaration:**

int IsCom3(void);

- **Input Parameter:**
  None.
- **Return Value:**
  QueueIsEmpty ..... QUEUE is empty
  QueueIsNotEmpty ..... QUEUE is not empty
  Others..... Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  None.

1.17 IsCom4

- **Description:**
  Check is there any data in the com4 input queue buffer .
- **Declaration:**
  int IsCom4(void);
- **Input Parameter:**
  None.
- **Return Value:**
  QueueIsEmpty ..... QUEUE is empty
  QueueIsNotEmpty ..... QUEUE is not empty
  Others..... Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  None.

1.18 ReadCom

- **Description:**
  Read one character from the input QUEUE buffer. If the QUEUE buffer is empty will return an ERROR code.
- **Declaration:**
  int ReadCom(int port);
- **Input Parameter:**

Argument	Descriptions
port	1/2/3/4 for COM 1/2/3/4

- **Return Value:**
  positive: → character read
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program**
  None.

1.19 ReadCom1

- **Description:**
  Read one character from COM1 input QUEUE buffer. If the QUEUE buffer is empty will return an ERROR code.
- **Declaration:**
  int ReadCom1(void);
- **Input Parameter:**
  None
- **Return Value:**
  positive: → character read
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program**
  None.

1.20 ReadCom2

- **Description:**
  Read one character from COM2 input QUEUE buffer. If the QUEUE buffer is empty will return an ERROR code.
- **Declaration:**
  int ReadCom2(void);
- **Input Parameter:**
  None
- **Return Value:**

positive: → character read
Others: → Error code(negative value, Refer to Sec. 1.1)

- **Demo program**
  None.

1.21 ReadCom3

- **Description:**
  Read one character from COM3 input QUEUE buffer. If the QUEUE buffer is empty will return an ERROR code.
- **Declaration:**
  int ReadCom3(void);
- **Input Parameter:**
  None
- **Return Value:**
  positive: → character read
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program**
  None.

1.22 ReadCom4

- **Description:**
  Read one character from COM4 input QUEUE buffer. If the QUEUE buffer is empty will return an ERROR code.
- **Declaration:**
  int ReadCom4(void);
- **Input Parameter:**
  None
- **Return Value:**
  positive: → character read
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program**

None.

1.23 ToCom

- **Description:**
  Transmit one character to Com port.
- **Declaration:**
  int ToCom(int port, char data);
- **Input Parameter:**

Argument	Descriptions
port	1/2/3/4 for COM 1/2/3/4
data	8-bit character to transmit

- **Return Value:**
  NoError: → OK
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  None.

1.24 ToCom1

- **Description:**
  Transmit one character to Com1.
- **Declaration:**
  int ToCom1(char data);
- **Input Parameter:**

Argument	Descriptions
data	8-bit character to transmit

- **Return Value:**
  NoError: → OK
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  None.

1.25 ToCom2

- **Description:**
  Transmit one character to Com2.
- **Declaration:**
  int ToCom2(char data);
- **Input Parameter:**

Argument	Descriptions
data	8-bit character to transmit

- **Return Value:**
  NoError: → OK
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  None.

1.26 ToCom3

- **Description:**
  Transmit one character to Com3.
- **Declaration:**
  int ToCom3(char data);
- **Input Parameter:**

Argument	Descriptions
data	8-bit character to transmit

- **Return Value:**
  NoError: → OK
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  None.

1.27 ToCom4

- **Description:**
  Transmit one character to Com4.
- **Declaration:**
  int ToCom4(char data);
- **Input Parameter:**

Argument	Descriptions
data	8-bit character to transmit

- **Return Value:**
  NoError: → OK
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  None.

1.28 ToComBufn

- **Description:**
  Send n bytes data to comport.
- **Declaration:**
  int ToComBufn(int port, char *buf, int no);
- **Input Parameter:**

Argument	Descriptions
port	1/2/3/4 for COM 1/2/3/4
buf	Pointer point to where the data stored.
no	The number of data to be sent.

- **Return Value:**
  NoError: → OK
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  None

1.29 ToCom1Bufn

- **Description:**
  Send n bytes data to COM1.
- **Declaration:**
  int ToCom1Bufn(char *buf, int no);
- **Input Parameter:**

Argument	Descriptions
buf	Pointer point to where the data stored.
no	The number of data to be sent.

- **Return Value:**
  NoError: → OK
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  None

1.30 ToCom2Bufn

- **Description:**
  Send n bytes data to COM2.
- **Declaration:**
  int ToCom2Bufn(char *buf, int no);
- **Input Parameter:**

Argument	Descriptions
Buf	Pointer point to where the data stored.
No	The number of data to be sent.

- **Return Value:**
  NoError: → OK
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  None

1.31 ToCom3Bufn

- **Description:**
  Send n bytes data to COM3.
- **Declaration:**
  int ToCom3Bufn(char *buf, int no);
- **Input Parameter:**

Argument	Descriptions
buf	Pointer point to where the data stored.
no	The number of data to be sent.

- **Return Value:**
  NoError: → OK
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  None

1.32 ToCom4Bufn

- **Description:**
  Send n bytes data to COM4.
- **Declaration:**
  int ToCom4Bufn(char *buf, int no);
- **Input Parameter:**

Argument	Descriptions
buf	Pointer point to where the data stored.
no	The number of data to be sent.

- **Return Value:**
  NoError: → OK
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  None

1.33 ToComStr

- **Description:**
  Send a string to comport.
- **Declaration:**
  int ToComStr(int port, char *str);
- **Input Parameter:**

Argument	Descriptions
port	1/2/3/4 for COM 1/2/3/4
str	Pointer point to where the data stored.

- **Return Value:**
  NoError: → OK
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  None

1.34 ToCom1Str

- **Description:**
  Send a string to COM1.
- **Declaration:**
  int ToCom1Str(char *str);
- **Input Parameter:**

Argument	Descriptions
str	Pointer point to where the data stored.

- **Return Value:**
  NoError: → OK
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  None

1.35 ToCom2Str

- **Description:**
  Send a string to COM2.
- **Declaration:**
  int ToCom2Str(char *str);
- **Input Parameter:**

Argument	Descriptions
str	Pointer point to where the data stored.

- **Return Value:**
  NoError: → OK
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  None

1.36 ToCom3Str

- **Description:**
  Send a string to COM3.
- **Declaration:**
  int ToCom3Str(char *str);
- **Input Parameter:**

Argument	Descriptions
str	Pointer point to where the data stored.

- **Return Value:**
  NoError: → OK
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  None

1.37 ToCom4Str

- **Description:**
  Send a string to COM4.
- **Declaration:**
  int ToCom4Str(char *str);
- **Input Parameter:**

Argument	Descriptions
str	Pointer point to where the data stored.

- **Return Value:**
  NoError: → OK
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  None

1.38 ClearCom

- **Description:**
  Clear the input QUEUE buffer of comport .
- **Declaration:**
  int ClearCom(int port);
- **Input Parameter:**

Argument	Descriptions
port	1/2/3/4 for COM 1/2/3/4

- **Return Value:**
  NoError: → OK
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  none

1.39 ClearCom1

- **Description:**
  Clear the input QUEUE buffer of COM1 .
- **Declaration:**
  int ClearCom1(void);
- **Input Parameter:**
  none
- **Return Value:**
  NoError: → OK
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  none

1.40 ClearCom2

- **Description:**
  Clear the input QUEUE buffer of COM2 .
- **Declaration:**
  int ClearCom2(void);
- **Input Parameter:**
  none
- **Return Value:**
  NoError: → OK
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  none

1.41 ClearCom3

- **Description:**
  Clear the input QUEUE buffer of COM3 .

- **Declaration:**
  int ClearCom3(void);
- **Input Parameter:**
  none
- **Return Value:**
  NoError: → OK
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  none

1.42 ClearCom4

- **Description:**
  Clear the input QUEUE buffer of COM4 .
- **Declaration:**
  int ClearCom4(void);
- **Input Parameter:**
  none
- **Return Value:**
  NoError: → OK
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  none

1.43 WaitTransmitOver

- **Description:**
  The COM1 & COM2 of 7188 can be used as RS-485 Com port. The RS-485 COM port of 7188 is 2-wire & half-duplex. Therefore the user has to control the direction of RS-485 for transmit/receive data. This function is used to detect the transmit over status, then the program can change the RS-485 from transmit-direction to receive-direction.

- **Declaration:**
  int WaitTransmitOver(int port);
- **Input Parameter:**

Argument	Descriptions
port	1/2/3/4 for COM 1/2/3/4

- **Return Value:**
  NoError: → transmit over
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  demo3.c

1.44 WaitTransmitOver1

- **Description:**
  The COM1 of 7188 can be used as RS-485 Com port. The RS-485 COM port of 7188 is 2-wire & half-duplex. Therefore the user has to control the direction of RS-485 for transmit/receive data. This function is used to detect the transmit over status, then the program can change the RS-485 from transmit-direction to receive-direction.
- **Declaration:**
  int WaitTransmitOver1(void);
- **Input Parameter:**
  none
- **Return Value:**
  NoError: → transmit over
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  none.

1.45 WaitTransmitOver2

- **Description:**
  The COM2 of 7188 is a 2-wire & half-duplex RS-485 Com port.

Therefore the user has to control the direction of RS-485 for transmit/receive data. This function is used to detect the transmit over status, then the program can change the RS-485 from transmit-direction to receive-direction.

- **Declaration:**
  int WaitTransmitOver2(void);
- **Input Parameter:**
  none
- **Return Value:**
  NoError: → transmit over
  Others: → Error code(negative value, Refer to Sec. 1.1)
- Demo program:
  none.

1.46 WaitTransmitOver3

- **Description:**
  The COM3 of 7188 is a RS-232 port. RS-232 is full-duplex so user need not to control the direction of RS-232. This function is used to detect the transmit over status, then the program can know the data has beed sent out.
- **Declaration:**
  int WaitTransmitOver3(void);
- **Input Parameter:**
  none
- **Return Value:**
  NoError: → transmit over
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  none.

1.47 WaitTransmitOver4

- **Description:**
  The COM4 of 7188 is a RS-232 port. RS-232 is full-duplex so user need not to control the direction of RS-232. This function is used to detect the transmit over status, then the program can know the data has beed sent out.
- **Declaration:**
  int WaitTransmitOver4(void);
- **Input Parameter:**
  none
- **Return Value:**
  NoError: → transmit over
  Others: → Error code(negative value, Refer to Sec. 1.1)
- Demo program:
  none.

1.48 Set485DirToTransmit

- **Description:**
  The COM1 & COM2 of 7188 can be used as RS-485 Com port. The RS-485 COM port of 7188 is 2-wire & half-duplex. Therefore the user has to control the direction of RS-485 for transmit/receive data. This function is used to change the RS-485 to transmit-direction.
- **Declaration:**
  int Set485DirToTransmit(int port);
- **Input Parameter:**
  port: → 1/2 for COM 1/2
- **Return Value:**
  NoError: → OK
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  demo3.c

1.49 Set485DirToReceive

- **Description:**
  The COM1 & COM2 of 7188 can be used as RS-485 Com port. The RS-485 COM port of 7188 is 2-wire & half-duplex. Therefore the user has to control the direction of RS-485 for transmit/receive data. This function is used to change the RS-485 to receive-direction.
- **Declaration:**
  int Set485DirToReceive(int port);
- **Input Parameter:**
  port: → 1/2 for COM 1/2
- **Return Value:**
  NoError: → OK
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo Program:**
  demo3.c

1.50 IsCom3OutBufEmpty

- **Description:**
  Check COM3 output buffer is empty or not. COM3 also use interrupt driver to send data out, user call **ToCom3** just store the data on output buffer, and the interrupt routine will send it out, user can call this function to check if all the data in output buffer is send out.
- **Declaration:**
  int IsCom3OutBufEmpty(void);
- **Input Parameter:**
  None.
- **Return Value:**
  When COM3 output buffer is empty return 1, else return 0
- **Demo program:**
  None.

1.51 IsCom4OutBufEmpty

- **Description:**
  Check COM4 output buffer is empty or not. COM4 also use interrupt driver to send data out, user call **ToCom4** just store the data on output buffer, and the interrupt routine will send it out, user can call this function to check if all the data in output buffer is send out.
- **Declaration:**
  int IsCom4OutBufEmpty(void);
- **Input Parameter:**
  None.
- **Return Value:**
  When COM4 output buffer is empty return 1, else return 0
- **Demo program:**
  None.

1.52 DataSizeInCom

- **Description:**
  Get the data number in comport input buffer.
- **Declaration:**
  int DataSizeInCom(int port);
- **Input Parameter:**

Argument	What it does
port	1/2/3/4 for COM1/2/3/4

- **Return Value:**
  the data number in comport input buffer.
- **Demo program:**
  none.

1.53 DataSizeInCom1

- **Description:**
  Get the data number in com1 input buffer.
- **Declaration:**
  int DataSizeInCom1(void);
- **Input Parameter:**
  None.
- **Return Value:**
  the data number in com1 input buffer.
- **Demo program:**
  none.

1.54 DataSizeInCom2

- **Description:**
  Get the data number in com2 input buffer.
- **Declaration:**
  int DataSizeInCom2(void);
- **Input Parameter:**
  None.
- **Return Value:**
  the data number in com2 input buffer.
- **Demo program:**
  none.

1.55 DataSizeInCom3

- **Description:**
  Get the data number in com3 input buffer.
- **Declaration:**
  int DataSizeInCom3(void);

- **Input Parameter:**
  None.
- **Return Value:**
  the data number in com3 input buffer.
- **Demo program:**
  none.

1.56 DataSizeInCom4

- **Description:**
  Get the data number in com4 input buffer.
- **Declaration:**
  int DataSizeInCom4(void);
- **Input Parameter:**
  None.
- **Return Value:**
  the data number in com4 input buffer.
- **Demo program:**
  none.

1.57 SendCmdTo7000

- **Description:**
  This function is use to send command to 7000's modules.
- **Declaration:**
  int SendCmdTo7000(int iPort, unsigned char *cCmd, int iChksum);
- **Input Parameter:**

Argument	What it does
port	1 to 4 for COM1 to COM4
cCmd	The address of the command want to be send.The command need not end with the command terminator 0x0d, This function will auto send 0x0d out after send the command and check sum(if check sum is enable).

Argument	What it does
iChksum	0: check sum disable, 1: check sum enable. If check sum is set to enable, the function will add 2 bytes check sum after the command.

- **Return Value:**
  When success return NoError.
  Others: return Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  demo12.c

1.58 ReceiveResponseFrom7000

- **Description:**
  This function use to receive response message from 7000's modules. The response message will end with 0x0D. If check sum is enable, the function will check it and remove checksum from the message. This function also remove the terminator code 0x0d.
- **Declaration:**
  int ReceiveResponseFrom7000(int iPort, unsigned char *cCmd, long lTimeout, int iChksum);
- **Input Parameter:**

Argument	What it does
iPort	1 to 4 for COM1 to COM4
cCmd	The address of buffer to save the received message.
lTimeout	This function will check the comport for receive message, if it check *lTimeout* times, and do not get a response message(end with 0x0d), it will return ***TimeOut*** error.
iChecksum	0: check sum disable, 1: check sum enable.

- **Return Value:**
  When success return NoError.
  Others: return Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  demo12.c

1.59 LedOff

- **Description:**
  Turn the red-LED off.
- **Syntax:**
  void LedOff(void);
- **Input Parameter:**
  None.
- **Return Value:**
  None.
- **Demo program:**
  demo1.c

1.60 LedOn

- **Description:**
  Turn the red-LED on.
- **Syntax:**
  void LedOn(void);
- **Input Parameter:**
  None.
- **Return Value:**
  None.
- **Demo program:**
  demo1.c

1.61 Init5DigitLed

- **Description:**
  Initial the hardware of 5-digit LED & blank all digits.
- **Declaration:**
  void Init5DigitLed(void);

- **Input Parameter:**
  None
- **Return Value:**
  None.
- **Demo program:**
  demo1.c

1.62 Show5DigitLed

- **Description:**
  Show a character to 5-digit LED.
- **Declaration:**
  int Show5DigitLed(int pos, int value);
- **Input Parameter:**

Argument	What it does
pos	1-5 for position 1-5(from left to right)
value	0 - 9 for show ‘0’-‘9’ 10 show ’A’ 11 show ’b’ 12 show ’c’ 13 show ’d’ 14 show ’e’ 15 show ’F’ 16 show ‘ ‘(blank or space) 17 show ’-‘ 18 show ‘.’(dot)

- **Return Value:**
- When success return NoError.
  Others: return Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  demo1.c , qc4.c

1.63 Show5DigitLedWithDot

- **Description:**
  Show a character to 5-digit LED. Show5DigitLedWithDot show the same character with Show5DigitLed and also show the dot.
- **Declaration:**
  int Show5DigitLedWithDot(int pos, int data);
- **Input Parameter:**

Argument	What it does
pos	1-5 for position 1-5(from left to right)
value	0 - 9 for show '0'-'9' 10 show 'A' 11 show 'b' 12 show 'c' 13 show 'd' 14 show 'e' 15 show 'F' 16 show ' '(blank or space) 17 show '-' 18 show '.'(dot) 0-17 will also show the dot('.')

- **Return Value:**
  When success return NoError.
  Others: return Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  qc4.c

1.64 Show5DigitLedSeg

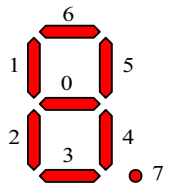
- **Description:**
  Show a seven-segment pattern to 5-digit LED. Use this function user can control every segment and the dot of 5-digit LED.

- **Declaration:**

int Show5DigitLedSeg(int pos, unsigned char data);

- **Input Parameter:**

Argument	What it does
pos	1-5 for position 1-5(from left to right)
value	Each bit of value control one segment of 7-segment led and the Dot. (see the following picture.) the bit is 1 will let the segment light, the bit is 0 will dark the segment. For example we want to show "1.", so must light segment 4,5,and 7, the value must be 0xB0.



- **Return Value:**
  NoError: → OK
  Others: → Error code(negative value, Refer to Sec. 1.1)

1.65 Set5DigitLedTestMode

- **Description:**
  This function can set 5-Digit LED into test mode(mode=1), all segment will light, and can set back to normal mode(mode=0).
- **Declaration:**
  void Set5DigitLedTestMode(int mode);
- **Input Parameter:**

Argument	What it does
mode	0: normal mode. 1: set to test mode. On test mode all the segment will light.

- **Return Value:**
  None
- **Demo program:**
  qc4.c

1.66 Set5DigitLedIntensity

- **Description:**
  This function can set the intensity of 5-Digit LED.
- **Declaration:**
  void Set5DigitLedIntensity(int mode);
- **Input Parameter:**

Argument	What it does
mode	0-15. Whem call Init5DigitLed the intensity will set to 7. Mode=15 is the most bright, mode=0 is the most dark.

- **Return Value:**
  When success return NoError.
  Others: return Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  qc4.c

1.67 Disable5DigitLed

- **Description:**
  Call **Disable5DigitLed** will blank all 5-Digit LED. Must call Enable5Digitled to back to normal mode.
- **Declaration:**
  void Disable5DigitLed(void);
- **Input Parameter:**
  None.
- **Return Value:**
  None.
- **Demo program:**
  QC4.c

1.68 Enable5DigitLed

- **Description:**
  After call **Disable5DigitLed** must call Enable5DigitLed if want set the 5-Digit LED back to normal mode.
- **Declaration:**
  void Enable5DigitLed(void);
- **Input Parameter**:
  None.
- **Return Value:**
  None.
- **Demo program:**
  QC4.C

1.69 ReadNVRAM

- **Description:**
  Read the content of the NVRAM. Total size of NVRAM is 31 bytes.
- **Declaration:**
  int ReadNVRAM(int addr);
- **Input Parameter:**

Argument	What it does
Addr	0 to 30

- **Return Value:**
  positive: → 8-bit NVRAM data
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  demo5.c

1.70 WriteNVRAM

- **Description:**

Write one byte data to NVRAM.

- **Declaration:**
  int WriteNVRAM(int addr, int data);
- **Input Parameter:**

Argument	What it does
Addr	0 to 30
Data	0 to 255(8-bit data)

- **Return Value:**
  NoError: → OK
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  demo5.c

1.71 WriteEEP

- **Description:**
  Write one byte data to EEPROM. EEPROM has 8 blocks, every block has 256 bytes, so total size is 2K bytes.
- **Declaration:**
  int WriteEEP(int block, int addr, int data);
- **Input Parameter:**

Argument	What it does
block	0 to 7(total 8 blocks)
addr	0 to 255(every block has 256 bytes)
data	0 to 255(8-bit data)

- **Return Value:**
  NoError: → OK
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  demo6.c

1.72 ReadEEP

- **Description:**
  Read one byte data from EEPROM.
- **Declaration:**
  int ReadEEP(int block, int addr);
- **Input Parameter:**

Argument	What it does
block	0 to 7(total 8 blocks)
addr	0 to 255(every block has 256 bytes)

- **Return Value:**
  Positive : →8-bit data read from EEPROM
  Others: → Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  demo6.c

1.73 EnableEEP

- **Description:**
  Enable the EEPROM for the write operation. The EEPROM is in the **write-protect state** since the first power on. The application program has to **enable the EEPROM before any write operation.** The EEPROM can be read in any condition.
- **Declaration:**
  void EnableEEP(void);
- **Input Parameter:** void
- **Return Value:** void
- **Demo program: demo6.c**
- **Example:**

```
EnableEEP();        /* enable before write to EEPROM */
WriteEEP(block,addr,data);
ProtectEEP();       /* protect EEPROM after write operation */
```

1.74 ProtectEEP

- **Description:**
  Protect the EEPROM to inhabit the write operation. The EEPROM is in the **write-protect state** since the first power on. The application program has to **enable the EEPROM before any write operation.** The EEPROM can be read in any condition.
- **Declaration:**
  void ProtectEEP(void);
- **Input Parameter:**
  None.
- **Return Value:**
  None.
- **Demo program:**
  demo6.c

1.75 FlashReadId

- **Description:**
  Use to check flash memory size. Is7188 will call this function to decide the flash memory size is 256K or 512K.
- **Declaration:**
  int FlashReadId(void);
- **Input Parameter:**
  None.
- **Return Value:**
  High byte return the flash type, and low byte return the manufacture number. 256K type will return 0xb001, 512K type will return 0xA401. The manufacture number 1 is AMD.
- **Demo program:**
  None

1.76 FlashWrite

- **Description:**
  When want save data to Flash memory just call this function and give the address(segment and offset) and data. Every time call FlashWrite just write one byte data to Flash memory. The data on flash memory only can be write from 1 to 0, if current value is 0xFF, that is every bit is 1, so it can be write to any value, if current value is 0x01 , only the bit 0 is 1, so only can write bit 0 to 0, that is only can write 0 to it, if want write other value to it, must erase flash memory first, erase flash memory will erase all the data bit to 1.
- **Declaration:**
  int FlashWrite(unsigned int seg, unsigned int offset, char data);
- **Input Parameter:**

Argument	What it does
seg	seg can be 0xc000 or 0xd000 od 0xe000 for 256K type, and can be 0x8000 to 0xe000 for 512K type.
offset	Range is 0 to 0xFFFF
data	Range is 0 to 0xFF

- **Return Value:**
  When success return NoError.
  Others: return Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  torom.c

1.77 FlashErase

- **Description:**
  FlashErase is used to erase a sector of flash memory. The sector size of Flash memory is 64K bytes, just map to a segment of 7188(D). The 256K type 7188 has three segments can be erase, these segments are 0xc000, 0xd000, 0xe000. The segment 0xF000 has bios and ROM-DOS so user must not erase it. The 512K type has the segments from 0x8000 to 0xe000 can be erased. In generally the first

segment of Flash memory is used by the ROM-DISK, so only when user want to download the new ROM-DISK image file will erase the first segment, for example TOROM.EXE will erase the first segment of Flash memory. After erase all the data on that Flash sector (segment) will be 0xFF, and then it can be write to any other value. When the value is write to another value, it must be erase again for write another value to it.

- **Declaration:**
  int FlashErase(unsigned int seg);
- **Input Parameter:**

Argument	What it does
seg	seg can be 0xc000 or 0xd000 od 0xe000 for 256K type, and can be 0x8000 to 0xe000 for 512K type.

- **Return Value:**
  When success return NoError.
  Others: return Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  torom.c

1.78 FlashRead

- **Description:**
  FalshRead is used to read one byte data from Flash Memory. This function is the same as the function **peekb** of TC(BC++). FlashRead also can read the memory locate on SRAM.
- **Declaration:**
  int FlashRead(unsigned int seg, unsigned int offset);
- **Input Parameter:**

Argument	What it does
seg	Segment of flash memory
offset	Offset of flash memory

- **Return Value:**
  Return the byte at the memory location addressed by seg:offset .
- **Demo program:**
  None.

1.79 EnableWDT

- **Description:**
  This function will enable the watchdog.
  The watchdog timer is fixed in 1.6 sec. If the watchdog is enable, the application program should refresh the watchdog timer before the 1.6 sec timer up.
  If the watchdog timer is up, the 7188 will be **reset** by the hardware circuit.
- **Declaration:**
  void EnableWDT(void);
- **Input Parameter:**
  None.
- **Return Value:**
  None.
- **Demo program:**
  demo7.c

1.80 RefreshWDT

- **Description:**
  This function is used to refresh the watchdog.
  The watchdog timer is fixed in 1.6 sec. If the watchdog is enable, the application program should refresh the watchdog timer before the 1.6 sec timer up.
  If the watchdog timer is up, the 7188 will be **reset** by the hardware circuit.
- **Declaration:**
  void RefreshWDT(void);
- **Input Parameter:**
  None.
- **Return Value:**
  None.
- **Demo program:**

None.

1.81 DisableWDT

- **Description:**
  This function can disable the watchdog.
  The watchdog timer is fixed in 1.6 sec. If the watchdog is enable, the application program should refresh the watchdog timer before the 1.6 sec timer up.
  If the watchdog timer is up, the 7188(D)/DOS will be **reset** by the hardware circuit.
- **Declaration:**
  void DisableWDT(void);
- **Input Parameter:**
  None.
- **Return Value:**
  None.
- **Demo program:**
  demo7.c

1.82 IsResetByWatchDogTimer

- **Description:**
  To check if 7188(D)/DOS is reset by. The function is used when 7188(D)/DOS is reset and the program is excute again, to check if the reset is according to WatchDog Timer power off or other reasons(reset by power off or just the program excute a far jmp to 0xFFFF:0000 or call int 19H).
- **Declaration:**
  int IsResetByWatchDogTimer(void);
- **Input Parameter:**
  None.
- **Return Value:**
  Return NoError (0) When 7188 is not reset by WatchDig Timer since

last time call **IsResetByWatchDogTimer**.
Return 1 When 7188 is reset by WatchDog Timer since last time call **IsResetByWatchDogTimer**. If call **IsResetByWatchDogTimer** again ,it will return 0.

- **Demo program:**
  None.

1.83 IsResetByPowerOff

- **Description:**
  To check if 7188(D)/DOS is reset by power off. The function is used when 7188(D)/DOS is reset and the program is excute again, to check if the reset is according to power off or other reasons(reset by watchdog timer or just the program excute a far jmp to 0xFFFF:0000 or call int 19H).
- **Declaration:**
  int IsResetByPowerOff(void);
- **Input Parameter:**
  None.
- **Return Value:**
  Return NoError (0) When 7188 is not reset by power off since last time call IsResetByPowerOff.
  Return 1 When 7188 is reset by power off since last time call IsResetByPowerOff. If call IsResetByPowerOff again ,it will return 0.
- **Demo program:**
  None.

1.84 TimerOpen

- **Description:**
  Install the timer function of 7188(D)/DOS. It support one timer ticks, 8 stopwatch timer and 8 countdown timer. Before use these timer functions must call **TimerOpen**.
- **Declaration:**

int TimerOpen(void);

- **Input Parameter:**
  none.
- **Return Value:**
  When install success return NoError.
  Return 1 if timer function is already installed
- **Demo program:**
  demo90.C.

1.85 TimerClose

- **Description:**
  Uninstall the timer function.
- **Declaration:**
  int TimerClose(void);
- **Input Parameter:**
- **Return Value:**
  Always return NoError.
- **Demo program:**
  demo90.C

1.86 TimerResetValue

- **Description:**
  Reset the timer tick value of timer to 0.
- **Declaration:**
  void TimerResetValue(void);
- **Input Parameter:**
  none.
- **Return Value:**
  none
- **Demo program:**
  demo90.c

1.87 TimerReadValue

- **Description:**
  Read the timer ticks, unit is ms. *Before call this function must call* ***TimerOpen*** *to install Timer drive*r.
- **Declaration:**
  unsigned long TimerReadValue(void);
- **Input Parameter:**
  none.
- **Return Value:**
  Return the current value of timer ticks
- **Demo program:**
  demo90.c

1.88 StopWatchReset

- **Description:**
  Reset the stopwatch value. The timer driver support 8 StopWatch.
- **Declaration:**
  int StopWatchReset(int channel);
- **Input Parameter:**

Argument	What it does
channel	The range is 0-7

- **Return Value:**
  When success return NoError.
  Others: return Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  none.

1.89 StopWatchStart

- **Description:**
  Set the StopWatch start to count time from 0. After this setting, the StopWatch timer will increment 1 every 1 ms.
- **Declaration:**
  int StopWatchStart(int channel);
- **Input Parameter:**

Argument	What it does
channel	The range is 0-7

- **Return Value:**
  When success return NoError.
  Others: return Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  demo92.c

1.90 StopWatchStop

- **Description:**
  After call **StopWatchStart**, when need not to use it,call this function to stop use the StopWatch Timer.
- **Declaration:**
  int StopWatchStop(int channel);
- **Input Parameter:**

Argument	What it does
channel	The range is 0-7

- **Return Value:**
  When success return NoError.
  Others: return Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  none.

1.91 StopWatchPause

- **Description:**
  Pause the StopWatch timer.
- **Declaration:**
  int StopWatchPause(int channel);
- **Input Parameter:**

Argument	What it does
channel	The range is 0-7

- **Return Value:**
  When success return NoError.
  Others: return Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  none.

1.92 StopWatchContinue

- **Description:**
  After call **StopWatchPause**, use this function to continue the StopWatch Timer.
- **Declaration:**
  int StopWatchContinue(int channel);
- **Input Parameter:**

Argument	What it does
channel	The range is 0-7

- **Return Value:**
  When success return NoError.
  Others: return Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  none.

1.93 StopWatchReadValue

- **Description:**
  Read the current value of StopWatch timer, the time unit is 1 ms.
- **Declaration:**
  int StopWatchReadValue(int channel,unsigned long *value);
- **Input Parameter:**

Argument	What it does
channel	The range is 0-7
value	A pointer to a unsigned long variable, to store the timer value.

- **Return Value:**
  When success return NoError. The timer value is store to the address **value**.
  Others: return Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  demo92.c

1.94 CountDownTimerStart

- **Description:**
  Start to use a CountDown timer. Call this function must set a start value for the CountDown timer to count down until it reach 0, that is "time is up".
- **Declaration:**
  int CountDownTimerStart(int channel,unsigned long count);
- **Input Parameter:**

Argument	What it does
channel	The range is 0-7
count	The start value of countdown timer

- **Return Value:**
  When success return NoError.
  Others: return Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**

demo91.c

1.95 CountDownTimerReadValue

- **Description:**
  Read the current value of CountDown timer.
- **Declaration:**
  int CountDownTimerReadValue(int channel,unsigned long *value);
- **Input Parameter:**

Argument	What it does
channel	The range is 0-7
value	A pointer to a unsigned long variable, to store the timer value.

- **Return Value:**
  When success return NoError. The timer value is store to the address **value**.
  Others: return Error code(negative value, Refer to Sec. 1.1)
- **Demo program:**
  demo91.c

1.96 InstallUserTimer

- **Description:**
  The timer will generate interrupt every 1 ms, user can install a timer function, then the timer interrupt will call the user function, that is the user timer function will be called every 1 ms. So the user function must be finished before 1ms.
- **Declaration:**
  void InstallUserTimer(void (*fun)(void));
- **Input Parameter:**

Argument	What it does
fun	The function pointer of user timer function.

- **Return Value:**

None.

- **Demo program:**
  Demo95.C

1.97 InstallUserTimer1C

- **Description:**
  The system timer will generate interrupt 8 every 1/18.2 second, and interrupt 8 service routine will call int 0x1C, user can install a user timer1C function, so when int 0x1c is called, it will call user timer1C function.
- **Declaration:**
  void InstallUserTimer1C(void (*fun)(void));
- **Input Parameter:**

Argument	What it does
fun	The function pointer of user timer1C function.

- **Return Value:**
  None.
- **Demo program:**
  None.

1.98 DelayTimeMs

- **Description:**
  Delay a period of time, unit 1ms. This function use software for loop to delay, so it is not so accurate, it is recommend to use **DelayMs** instead of **DelayTimeMs**.
- **Declaration:**
  void DelayTimeMs(unsigned int time);
- **Input Parameter:**

Argument	What it does
time	The time interval want to delay, unit is 1 ms .

- **Return Value:**

void

- **Demo program:**
  demo1.c

1.99 DelayMs

- **Description:**
  Delay a period of time. The time unit is 1 ms . TC or BC++ also has a function **delay** , but it can not be use on 7188(D)/DOS. When use timer function, that is after call **TimerOpen** don't call **Delay**, please use **DelayMs** instead. When **TimerOpen** is not called, **DelayMs** will call **Delay**. It is recommend to use **DelayMs** instead of **Delay**.
- **Declaration:**
  void DelayMs(unsigned ms);
- **Input Parameter:**

Argument	What it does
ms	The time interval want to delay, unit is 1 ms .

- **Return Value:**
  none
- **Demo program:**
  none

1.100 Delay

- **Description:**
  Delay a period of time. The time unit is 1ms. TC or BC++ also has a function **delay** , but it can not be use on 7188(D)/DOS. When use timer function, that is after call **TimerOpen** , don't call **Delay**, please use **DelayMs** instead. It is recommend to use **DelayMs** instead of **Delay**.
- **Declaration:**
  void Delay(unsigned ms);
- **Input Parameter:**

Argument	What it does
ms	The time interval want to delay, unit is 1 ms .

- **Return Value:**
  none
- **Demo program:**
  none

1.101 Delay_1

- **Description:**
  Delay a period of time. The time unit is 0.1ms .
- **Declaration:**
  void Delay_1(unsigned ms);
- **Input Parameter:**

Argument	What it does
ms	The time interval want to delay, unit is 0.1 ms .

- **Return Value:**
  none.
- **Demo program:**
  none.

1.102 getch4

- **Description:**
  Get a key input from keyboard buffer.
- **Declaration:**
  int getch4(void);
- **Input Parameter:**
  none.
- **Return Value:**
  the next key input value.(the key is received from COM4 of 7188)
- **Demo program:**
  echo485.c

1.103 kbhit4

- **Description:**
  Check the keyboard buffer if a key input is currently available. **kbhit4** is used to instead of kbhit .
- **Declaration:**
  int kbhit4(void);
- **Input Parameter:**
  none.
- **Return Value:**
  return 0 when keyborad buffer is empty. Others when keyboard buffer is not empty.
- **Demo program:**
  echo485.C

1.104 ungetch4

- **Description:**
  Put a key code back to keyboard buffer for latter to read input from keyboard will read the key. **ungetch4** call int 0x16 function AH=5.
- **Declaration:**
  int ungetch4(int key);
- **Input Parameter:**

Argument	What it does
key	The key value want to putback to keyboard buffer.

- **Return Value:**
  When success return NoError.
  Others: keybiard buffer is full
- **Demo program:**
  none

1.105 putch4

- **Description:**
  Send one byte data to COM4. **putch4** does not call int 0x21, it direct call int 0x10 function 0x0E .
- **Declaration:**
  void putch4(int data);
- **Input Parameter:**

Argument	What it does
data	The value want to send out from COM4 (only the low byte was send out)

- **Return Value:**
  none
- **Demo program:**
  none

1.106 InitLib

- **Description:**
  On early version of 7188(D)/DOS library, **InitLib** will initialize the software driver. Since the version 1.05, InitLib do nothing, so now user need not to call this function at the beginning of a 7188 program. But it is better to call it for good compatible with early or later version of library.
- **Declaration:**
  void InitLib(void);
- **Input Parameter:**
  none
- **Return Value:**
  none
- **Demo program:**
  demo2.c

1.107 GetLibVersion

- **Description:**
  Get the version of 7188(D)/DOS library.
- **Declaration:**
  unsigned GetLibVersion(void);
- **Input Parameter:**
  none
- **Return Value:**
  The high byte is the major version number, the low byte is the minor version number. For example version 1.12 will return 0x010C.
- **Demo program:**
  none

1.108 Is7188

- **Description:**
  Check if program is run on 7188(D)/DOS, if is run on 7188 the function also check the Flash Memory size.
- **Declaration:**
  int Is7188(void);
- **Input Parameter:**
  none
- **Return Value:**
  0 the program is not run on 7188(D)/DOS
  256 the program is run on 7188(D)/DOS, and the Flash is 256K
  512 the program is run on 7188(D)/DOS, and the Flash is 512K
- **Demo program:**
  TOROM.C

1.109 ReadInitPin

- **Description:**
  Read the status of INIT*-pin.
- **Declaration:**
  int ReadInitPin(int port);
- **Input Parameter:**
  none
- **Return Value:**
  InitPinIsOpen: → INIT*-pin is OPEN.
  InitPinIsNotOpen: → INIT*-pin is short to the GND-pin.
- **Demo program:**
  demo1.c

1.110 hex_to_ascii

- **Description:**
  hex_to_ascii is a character array,size is 16 bytes, for convert 0-15 to character '0'-'F'
- **Declaration:**
  extrern char hex_to_ascii[16];
- **Input Parameter:**
  index value: 0-15.
- **Return Value:**
  Hex digit '0'-'9' or 'A'-'F'
- **Demo program:**
  ECHO485.C

1.111 ascii_to_hex

- **Description:**
  Convert character '0','1',..,'A','B',..,'F' to value 0,1, .. ,10,11, .. , 15

- **Declaration:**
  int ascii_to_hex(char ascii);
- **Input Parameter:**

Argument	What it does
ascii	Ascii can be ‘0’-‘9’ or ‘A’-‘F’ or ‘a’-‘f’

- **Return Value:**
  return th econverted value 0-15.
- **Demo program:**
  none

1.112 _MK_FP

- **Description:**
  Makes a far pointer. If use TC or BC++, user can use MK_FP, but MSC not support the function, so can use _MK_FP to make a far pointer.
- **Declaration:**
  void far *_MK_FP(unsigned segment, unsigned offset);
- **Input Parameter:**

Argument	What it does
segment	The segment of the far pointer
offset	The offset of the far pointer

- **Return Value:**
  The far pointer segment:offset.
- **Demo program:**
  None