

# **Boot Loader**

## **A Key Technique for MCU Development**

# Monitor/Debug or Boot Loader

- A basic residential program in MCU
  - Initialize the MCU system
  - Communication with people
  - Download application program from outside and run
  - Some debug utilities

**It is a basic development tool for Hardware and Software**

# **A Minimum MCU System**

- **A minimum MCU system is the BASIC hardware tool for application development**
- **Basic hardware system includes:**
  - **Power supply**
  - **Reset circuit**
  - **Clock circuit**
  - **A communication port (in most case: RS-232)**

# Tools for Hardware Test

- **Power supply**                      **Multi-meter**
- **Rest**                                **Multi-meter**
- **Clock circuit**                    **Oscilloscope**
  - **Be careful with the PCB design, reserve a oscillator module on the PCB**
- **RS-232**                              **Multi-meter**  
   **Oscilloscope**
  - **Need a program to test:**
    - **Initial SCI**
    - **Send out a ASCII loop**

**Need a tool to download program (BDM) anyway**

# Monitor/Debug (Boot Loader)

- **Initialize the MCU system**
- **Communication with people**
- **Download application program from outside and run**
- **Some debug utilities**

# Initialize the MCU system

- Initial Stack Pointer SP

**LDS #STACK ; Init. Stack Pointer**

- Initial SCI
- SCI need system clock, which maybe complicated
  - Use external clock first
  - Then the PLL initial

# Initial SCI

```
SC_Init LDAA #$0C           ; Initial SCI
        STAA SCI0CR2        ; Enable T & R
        LDAA #$0            ; Use default Protocol
        STAA SCI0BDH        ; 8b, No parity, 1stop
        LDAA #$9C           ; 9600 baud rate (clock related)
        STAA SCI0BDL
```

May not need

SCI clock = baud rate x 16 = bus clock/divider

Bus clock is from external clock OR PLL!

# SCI Test and Debug

Write a program send: AAAAAAAAAAAAAAAAAAAAAA.....

OUTCH	LDB	#'A'	
OUTCH1	BRCLR	SCI0SR1,\$80,*	
	STAB	SCI0DRL	;Write to SCI
	BRA	OUTCH1	

Till now, we believe that **IT WORKS!**

**Only 9 lines ASM code!**

Then Replace the BRA to RTS



# Input char Test

\*

```
INCH    BRCLR SCI0SR1,$20,INCH ;Check status
        LDAB  SCI0DRL
        BSR   OUTCH
        BRA   INCH
```

Then Replace the BRA to RTS

# Get\_Char

```
INCH  BRCLR  SCI0SR1,$20,INCH  ; Without Echo
      LDAA   SCI0DRL
      RTS
```

\*

```
INCHE  BSR   INCH      ; With Echo
      BRA   OUTCH
```

# Assembler Example: OUTCH

```
OUTCH      BRCLR   SCI0SR1,$80,*    ; Check if TBF Empty?
           STAA    SCI0DRL          ; Out char.
           RTS
```

**\* Out space(s) Routine:**

```
OUT2S      BSR     OUT1S
OUT1S      LDAA    #$20             ;ASCII Code for Space ' '
           BRA     OUTCH
```

**\* Print string PDATA:**

```
PRINT      JSR     OUTCH
PDATA      LDAA    ,X+
           CMPA    #EOT
           BNE     PRINT
           RTS
```

# Basic Functions of the Debug

- Initialize the MCU system
- Communication with people (**Talk to CPU**)
- **Download program and run**
- **Some debug utilities**

# Dynamic and Static CPU Registers Image

- **During IRQ or SWI, CPU registers are pushed into stack automatically That is dynamic CPY Image.**
- **A data structure in RAM, which has the same structure with above, it is called static image**
- **Get Static Image :**
  - **Copy Dynamic to Static**
  - **Let SP→Dynamic, then run SWI**

# Stack Structure after SWI

After IRQ/SWI

SP=SP-9



	Stack free byte
SP - 9	CCR
SP - 8	B
SP - 7	A
SP - 6	XH
SP - 5	XL
SP - 4	YH
SP - 3	YL
SP - 2	PCH
SP - 1	PCL
SP	Last Valid Data in the stack

Before IRQ

SP



Low Address



Stack Growth

High Address

# Run Program

- In uC/OS-II a task's context switch (CPU registers) is in the stack
- Move SP to the stack top run RTI Instruction to resume the task
- If we move the SP and let it points to the static Image and Run TRI, what happens?

# Display & Modify the CPU Registers

- Insert **SWI** to break the running CPU anywhere you like
- Copy Images from the stack to Static Image
- Display the Image
- Modify the Image
- Run **RTI** to run the program

**RAM accesses !**



# Reserve Memory for the Image

**ORG \$3FD0**

<b>MRCCR</b>	<b>RMB 1</b>	<b>;SP --&gt; Initialed Here</b>
<b>MRB</b>	<b>RMB 1</b>	<b>;SP + 1</b>
<b>MRA</b>	<b>RMB 1</b>	<b>;SP + 2</b>
<b>MRX</b>	<b>RMB 2</b>	<b>;SP + 3</b>
<b>MRY</b>	<b>RMB 2</b>	<b>;SP + 5</b>
<b>MRPC</b>	<b>RMB 2</b>	<b>;SP + 7</b>
<b>SPBUF</b>	<b>RMB 2</b>	<b>;SP + 9</b>

See Page 374

# You can Use C (see: OS\_CPU\_C.C)

```
void *OSTaskStkInit (void (*task)(void *pd), void *pdata, void *ptos, INT16U opt)
{ INT16U *stk;
  stk  = (INT16U *)ptos;          // Load stack pointer
  *--stk = opt;                  // opt one byte blank
  *--stk = (INT16U)(task);       // PC for use of opt in task
  *--stk = (INT16U)(task);       // PC
  *--stk = (INT16U)(0x1122);     // Y
  *--stk = (INT16U)(0x3344);     // X
  ((INT8U *)stk)--;             // One byte for A
  *(INT8U *)stk = (INT8U)((((INT16U)pdata)>>8); // A
  ((INT8U *)stk)--;             // One byte for B
  *(INT8U *)stk = (INT8U)(pdata); // B
  ((INT8U *)stk)--;             // One byte for CCR
  *(INT8U *)stk = (INT8U)(0x00); // CCR
  ((INT8U *)stk)--;             // One byte for PPAGE
  *(INT8U *)stk = *(INT8U *)pdata; // PPAGE
  return ((void *)stk); }
```

# Run Program

GO	LDS	SPBUF	; Current SP
	LDX	MRPC	
	PSHX		
	LDX	MRY	
	PSHX		
	LSX	MRX	
	PSHX		
	LDA	MRB	
	PSHA		
	LDAA	MRCC	
	PSHA		
	RTI		

# Command Jump Table

JMPTAB	FCC	'E'
	FDB	ERSPLSH
	FCC	'D'
	FDB	MDUMP
	FCC	'G'
	FDB	GO
	FCC	'H'
	FDB	HELP
	FCC	'L'
	FDB	DWNLD
	FCC	'M'
	FDB	MEMCHG
	FCC	'R'
	FDB	REGDSP
	.....	
FCC	'Z'	
FDB	CLEARBP	
TBLEND	EQU	*

# Increment command One by One

NXTCMD JSR INCH

LDX #JMPTAB

NXTCHR CMPB 0,X ; According to the command

BNE NEXT ; Jump to the corresponding routine

LDX 1,X

JSR 0,X

BRA NXTCMD

NEXTCMD INX

INX

INX

CPX #TBLEND

BLT NXTCHR

LDX #MSG4 ;'WHAT?'

JSR PDATA

BRA NXTCMD

# Routine to Modify PC

- \*
- \* Alter "PC" the Program Counter
- \*

```
ALTPC      JSR  PRTPC      ; PC=  
            JSR  OUT1S  
            JSR  IN1ADR  
            BVS  ALTPC1  
            STX  MRPC  
ALTPC1     RTS
```

# Input an Address

\*

\* Input 16 Bits ADDR. in X and BUF0,BUF1

\*

```
IN1ADR      BSR  BYTE      ; A HEX ADDR. in X & BUF0
            BVS  IN1ADN
            STAA BUF0
            BSR  BYTE
            BVS  IN1ADN
            STAA BUF1
            LDX  BUF0
IN1ADN      RTS
```

# Input a Byte (2 HEX ASCII in A)

BYTE	BSR	INHEX
	BVS	BYTEN
	ASLA	
	ASLA	
	ASLA	
	ASLA	
	TAB	
	BSR	INHEX
	BVS	BYTEN
	ABA	
BYTEN	RTS	
INHEX	BSR	ECHOM
	CMPA	#\$30
	BCS	NOTHEX
	CMPA	#\$39
	BHI	INHEXA
	SUBA	#\$30
	RTS	

INHEXA	ANDA	#\$DF
	CMPA	#\$41
	BCS	NOTHEX
	CMPA	#\$46
	BHI	NOTHEX
	SUBA	#\$37
	RTS	
NOTHEX	SEV	
	RTS	



# Down Load .S19 File to RAM

S0**13**0000062696E2F4347656E657269632E616273**20**

S1**23C000**3BEE862006B765E6856B30E.....C5**D9**

S1**23C020**1DC7877C23747C23727B215..... . 21**4D**

S1**23C040**0D7C210B7C21097B21117C2..... . 8D**E8**

S9**03**00000FC

Read Page 123

# Set Break Points

- **There are break point registers available**
- **If not, use SWI to replace the Instruction at the break points**

# Protection Area in the FLASH

- To prevent erase the Monitor/Debug by accident, it should be put into the protected area.
- The area can be 2K,4K,6K,8K.....from the top of the FLASH
- The vector area is at the top of the FLASH, and also protected
- It should be move to User programmable area

# Move Vectors to User Area

```
V38      LDX      $EFD6
          PSHX
          RTS

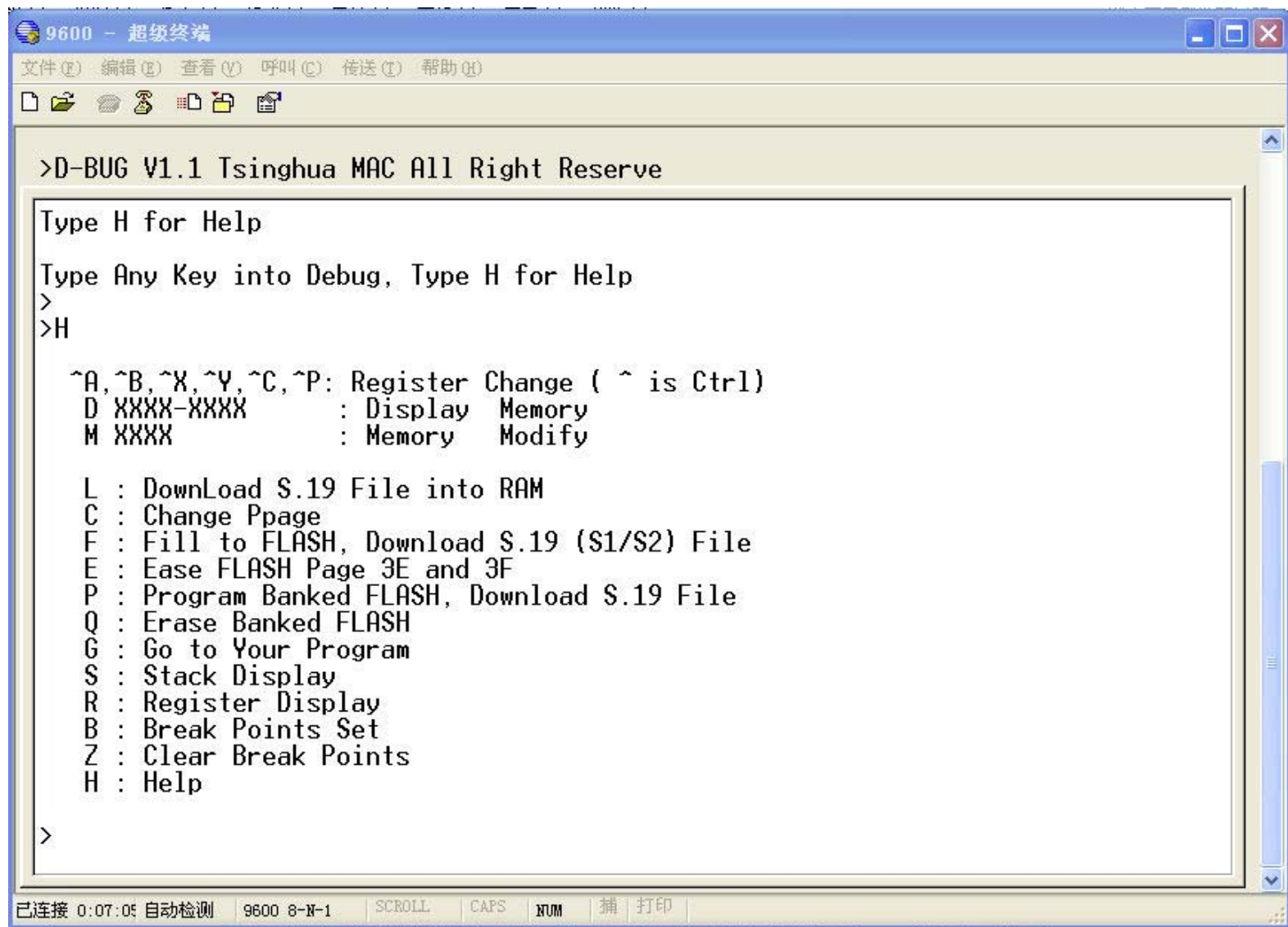
*
          ORG      $FF8C ; Vectors table

*
FF8C      FDB      V1      ; PWM Emergency Shutdown
          FDB      V2      ; Port P Interrupt
          FDB      V3      ; MSCAN 4 transmit
          .....
          FDB      V37     ; SCI 1
FFD6      FDB      V38     ; SCI 0
          FDB      V39     ; SPI0
          .....
          FDB      V56     ; COP failure reset
          FDB      V57     ; Clock Monitor fail reset
FFFE      FDB      V58     ; Reset
```

# System Call Routine

	ORG	\$FF40
COLDSTART	JMP	START
WARMSTART	JMP	NXTCMD
GETCHAR	JMP	INCH
PUTCHAR	JMP	OUTCH
PSTRING	JMP	PDATA
.....	.....	.....

# Help List



# Discussion

- How to do the Trace?