目录

1 绪论

2 单核处理器 8086/8088

单核处理器是多核处理器的基础
从 8086 开始 Intel CPU 设计采用了向后兼容，也称向下兼容
单核的 8086 CPU 成为其后 Intel CPU 的基石

2.1 8086/8088 处理器功能特性

1. 第一次将流水线思想引进微处理器：指令级流水
2. 存储器分段管理机制引入处理器，扩大寻址能力
3. 只有整数运算指令。配套数值协处理器 8087、输入输出协处理器 8089，具备较强大的计算能力和 I/O 处理能力

8086 CPU 有三个版本 8086、8086-2、8086-1。仅时钟频率不同，依次为 5Mhz、8Mhz、10Mhz。具有以下功能特性：

1. 直接主存寻址能力 1MB
2. 体系结构针对强大的汇编语言和有效的高级语言设计
3. 14 个 16 位寄存器
4. 24 种操作数寻址方式
5. 操作数类型：位、字节、字和块
6. 8、16 位无符号和带符号二进制和十进制运算，包括乘除

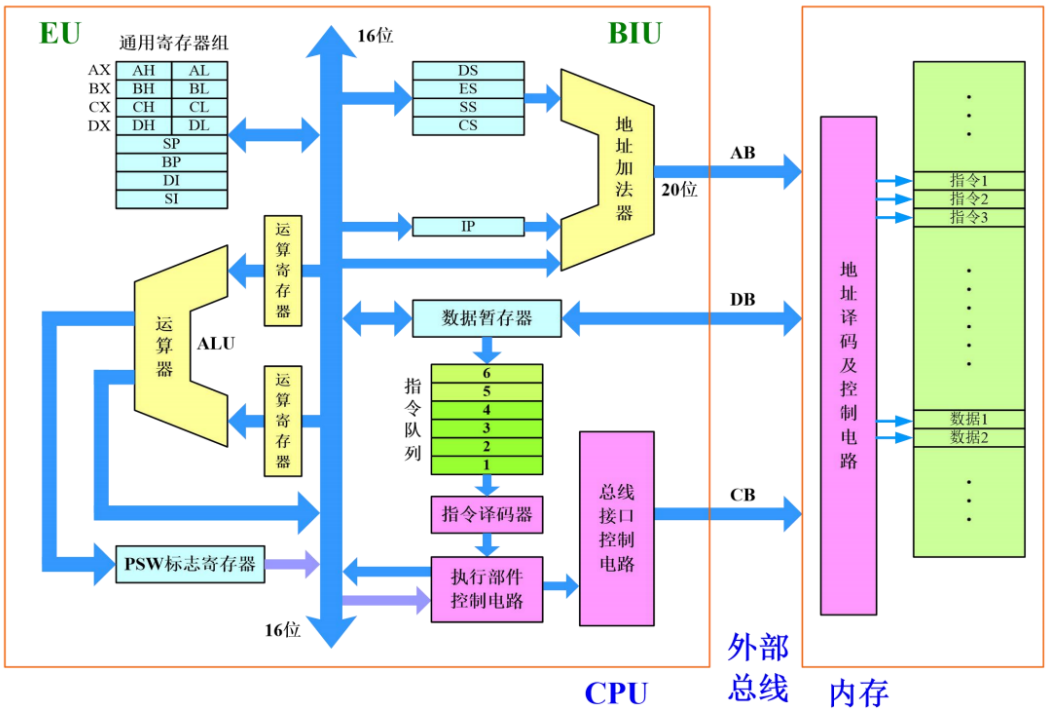

图 1: 8086 处理器体系结构

右侧的 BIU 是总线接口单元（Bus Interface Unit）负责与存储器、I/O 接口传递数据，具体完成：

1. 从内存中取指令、送到指令队列
2. 配合 EU 从指定的内存单元或 I/O 端口取数据
3. 将 EU 的操作结果送到指定的内存单元或 I/O 端口

左侧执行单元 EU 负责指令的执行（算术、逻辑、移位运算，有效地址计算，控制命令、……），包括 ALU，通用寄存器组和状态寄存器，进行 8/16 位运算。

取指令和执行指令在时间上重叠，BIU 在指令队列缓冲器有 2 个以上空字节时就不断从主存连续地址单元中取得指令送入指令队列缓冲器中，EU 则不断从指令队列中取出指令译码执行。仅当一下情况发生时，两者并行工作的状态打破：

1. 6 个字节的指令队列缓冲器满，且 EU 没有主存或 I/O 访问请求时，BIU 进入空闲状态

2. EU 执行访存或 I/O 指令时，需要对主存或 I/O 设备进行读写数据操作，则在 BIU 执行取指周期后，暂停取指操作，在下一总线执行 EU 所要求的主存或 I/O 读写操作，之后再继续 BIU 的取指操作。

3. 在 EU 执行转移、调用、返回等程序跳转指令时，因之前读入指令队列缓冲器的指令已无效，则 BIU 在清楚缓冲器的同时，根据 EU 提供的跳转地址，重新获取跳转后的程序指令。

工作原理：

1. DB 线中取数据到数据暂存器里

2. 指令队列有 2+ 空字节，则 BIU 自动取指，放到指令队列中

3. EU 总是从队列前部取指令执行

4. 指令要访问 M 或 I/O，EU 会请求 BIU 完成

优点：实现了预取指令，取指令和执行指令可以并行，加速程序运行。由于有指令队列，BIU 和 EU 可以并行工作，一边取指令、一边执行指令，构成指令级流水线。

2.2 寄存器、主存和 I/O 结构

2.2.1 寄存器

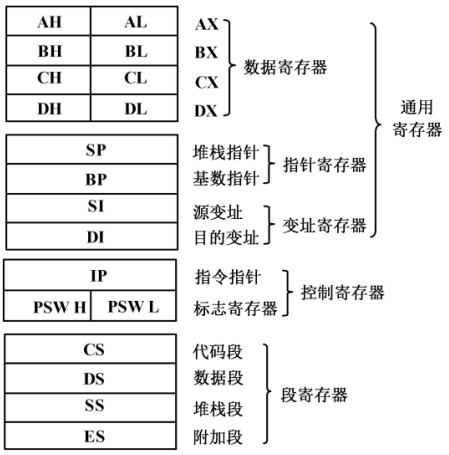

图 2: 寄存器结构

首先是数据寄存器:

1. 通用寄存器组均为 16 位，但是可以拆分成高位和低位转化成 8 位
2. AX 用于字乘法、字除法和字节乘法、字节除法以及字 I/O
3. BX 有时作为地址寄存器保存段内偏移地址
4. CX 有时作为隐含计数器串操作计数循环计数 CL 移位次数
5. DX 有时作为地址寄存器保存 I/O 地址

指针寄存器

1. SP 堆栈指针寄存器，用于存放主存堆栈区的偏移地址，指示当前操作位置
2. BP 基数指针寄存器，用于存放主存的基本偏移地址

变址寄存器

1. SI 源变址寄存器指向源操作数

2. DI 目的变址寄存器指向目的操作数

3. SI、DI 具有自动修改内容的功能

控制寄存器

1. IP 指令指针寄存器，用来指示指令所在存储单元的段内偏移地址。系统将欲执行程序的首地址加载至 CS 和 IP 中，转移类指令用跳转的目标地址修改 IP（或 CS 和 IP）。当以及 CS 和 IP 从主存取指令后 IP 自动加一，指向下一个读取的指令字节

2. PSW 是程序状态字，也称为状态寄存器或标志寄存器格式如下

图 3: 状态标志位的值

1. PSW 为 16 位，暂定 9 个标志位

2. C——进位标志。加减运算若在最高位出现进位或错位，则标志为 1，否则清 0. 逻辑运算、位移和循环指令也影响该标志位

3. P——奇偶标志位。低 8 位中 1 的个数位偶数时，标志位置位 1，否则清 0

4. A——半加进位标志位。加减运算时，低四位向高思维进位或者借位，则置 1，否则清 0。用于 BCD 运算结果矫正

5. Z——零标志位。运算结果全为 0，则为 1，否则清 0

6. T——陷阱标志位（单步标志位）。为 1 则进入单步执行指令工作方式，产生单步中断，CPU 执行陷阱中断程序。将执行结果显示（寄存器、存储单元和 I/O 接口）

7. I——中断允许标志位。为 1 则可屏蔽中断请求，为 0 则不可

8. D——方向标志位。为 1 则 SI 和 DI 在串操作指令执行中自动减即从高到低，否则 SI 和 DI 自动增量。

9. O——溢出标志位。当带符号数运算超出 8 位或 16 位表示范围

段寄存器

由于 8086 的设计，主存是按段划分，分为数据段、堆栈段、代码（程序）段、附加段。

1. 代码段寄存器 CS 指示程序区
2. 数据段寄存器 DS 指示数据区
3. 堆栈段寄存器 SS 指示堆栈区
4. 附加段寄存器 ES 指示数据区

1. SS:SP → 堆栈地址
2. CS:IP → 主存地址类似 PC
3. 首先找段地址寄存器 + 段内偏移地址
4. 地址加法器只做地址加法，输出为 I/O、存储器地址

2.2.2 主存结构

双体结构，既可以实现 16 位存储，也可以实现 8 位存储

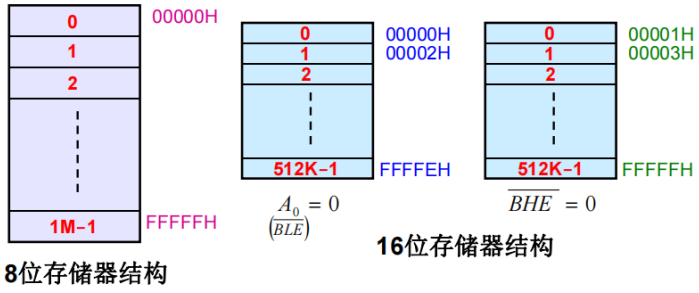

图 4: 双体结构

分段结构

1. 代码、数据量不大-> 使其处于同一段内（64KB 范围内）-> 减少指令长度、提高运行速度
2. 内存分段为程序的浮动分配创造条件
3. 形成地址 6832H:1280H-> 物理地址 = 68320H+1280
4. 各个分段之间可以重叠

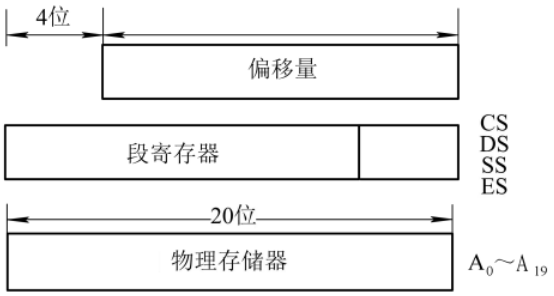

图 5: 物理地址形成

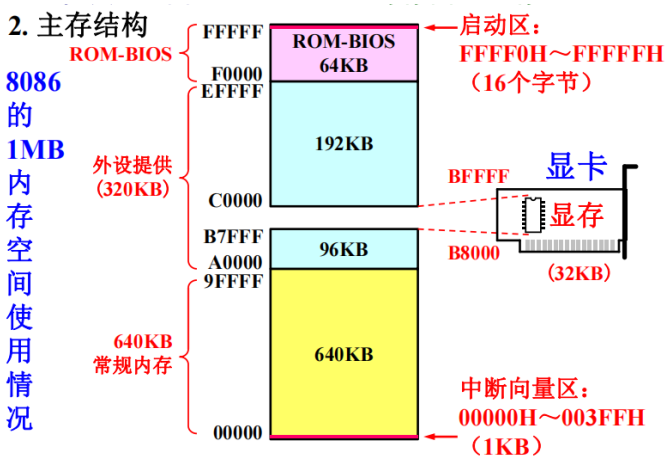

图 6: 主存结构

2.2.3 I/O 地址空间

BIU 处的地址加法器，将 16 位段地址左移 4 位，然后与 16 位段内偏移地址相加，生成 20 位物理地址，即

$$\text{物理地址} = \text{段地址} \times 16 + \text{段内偏移地址} \tag{1}$$

其中：段地址由 16 位段寄存器提供，段内偏移地址由 IP 或 EU 确定的有效地址 EA 提供

8086 处理器提供 20 位地址，对主存单元寻址时使用全部 20 位地址，对 I/O 设备端口寻址使用其低 16 位地址，使主存有 1M 的存储空间，I/O 设备有 64k 的端口空间（两个字节）

8086 采用字节编制，即主存或 I/O 的一个地址单元内存储一个字节，拥有一个字节编号，使得处理器可以根据地址进行 1 字节的存储器或 I/O 设备的读写操作

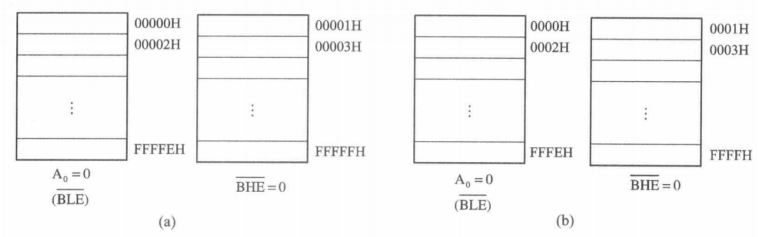

图 2.4　8086 系统的主存和 I/O 系统组织结构
(a) 主存储器结构；(b) I/O 结构

主存与 I/O 地址空间均按地址的奇偶分为两个体，偶地址空间对应低字节，8086 使用低字节允许信号 $A_0 = 0$ 选择；奇地址空间对应高字节的访问用高字节允许信号 $\overline{BHE} = 0$ 选择，具体选择如下：

$\overline{BHE}$	A_0	操 作 说 明
0	0	高字节体和低字节体同时有效，依据地址 n，从主存或 I/O 空间读/写 16 位数据(当 n 为偶地址时，仅需要 1 个总线周期就可以完成 2 字节的读/写，其中从地址 n 读/写数据低 8 位，从地址 n + 1 读/写数据高 8 位；当 n 为奇地址时，则需要 2 个总线周期才可以完成 2 字节的读/写，其中第一个总线周期从地址 n 读/写数据低 8 位，第二个总线周期从地址 n + 1 读/写数据高 8 位)
0	1	高字节体有效，依据奇地址 n，从主存或 I/O 空间的奇地址体读/写 8 位数据
1	0	低字节体有效，依据偶地址 n，从主存或 I/O 空间的偶地址体读/写 8 位数据
1	1	高字节体和低字节体无效，不能进行主存或 I/O 访问

图 7: 8086 CPU 允许信号的作用

8086 将 1MB 存储空间分为若干个存储段，每段固定 64KB 大小，用 16 位段地址寻址，每段的起始地址位 xxxx0000，低四位为 0。且由于段地址由 16 位段寄存器提供，所以 1MB 可以划分 2^{16} 个重叠段，不重叠有 2^4 个

分段结构使 16 位地址的寻址空间 64KB 扩大到 1MB，设置了 4 个属性的存储段：代码段、数据段、堆栈段和附加段。段寄存器 CS，DS，SS 和 ES 分别为 4 个属性段提供段地址。

1. 程序段存放指令代码
2. 数据段存放原始数据、中间结果和最终结果
3. 堆栈段用来设立堆栈

4. 好处是减少程序数据和堆栈的冲突

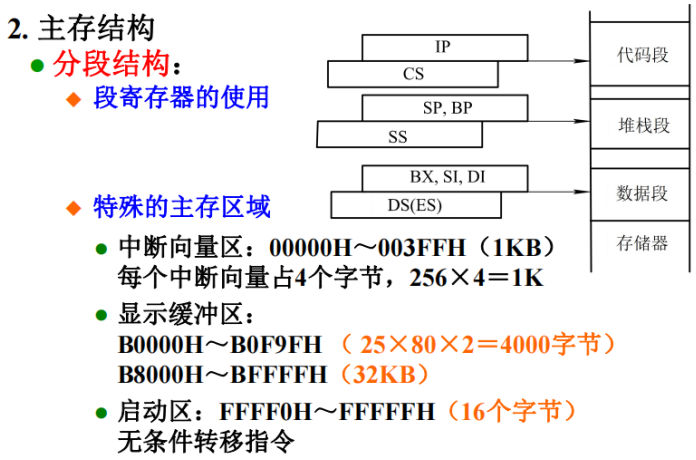

图 8: 分段结构

各类的访存操作中，段地址由默认或指定的段寄存器提供，CPU 根据指令中提供的段内偏移地址来源自动的默认使用段寄存器。

段寄存器 DS、SS 和 ES 的内容用传送指令加载，但是任何传送指令不能向代码段寄存器 CS 加载。JMP CALL RET INT IRET 等可以设置和影响 CS。无论程序去、数据区还是堆栈区都可以超过 64KB 的容量，都可以利用重新设置段寄存器内容的方法加以扩大

1. I/O 地址空间独立于主存地址空间，两者采用不同的读写信号进行访问控制

2. I/O 地址空间包含 64k 个可单独寻址的 8 位 I/O 端口，编号 0 到 FFFFH，其中 I/O 端口地址 0F8H 0FFH 被保留

3. 16 位系统 I/O 也按地址的奇偶分为个体

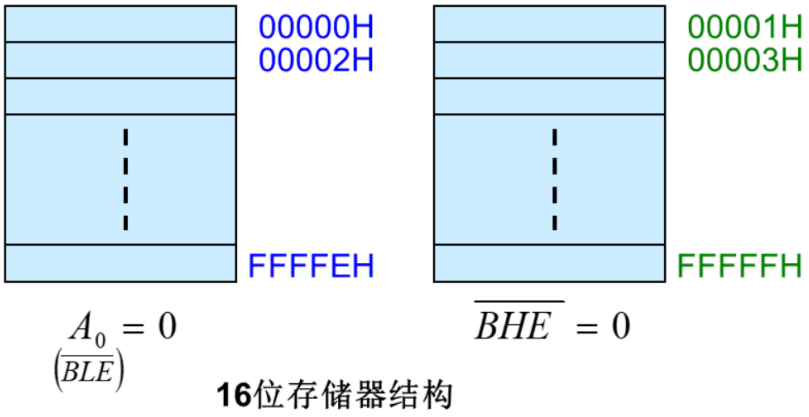
00000H
00002H
FFFFEH
00001H
00003H
FFFFFH
$A_0 = 0$
$(\overline{BLE})$
$\overline{BHE} = 0$
16位存储器结构

图 9: I/O 结构

2.2.4 处理器芯片引脚

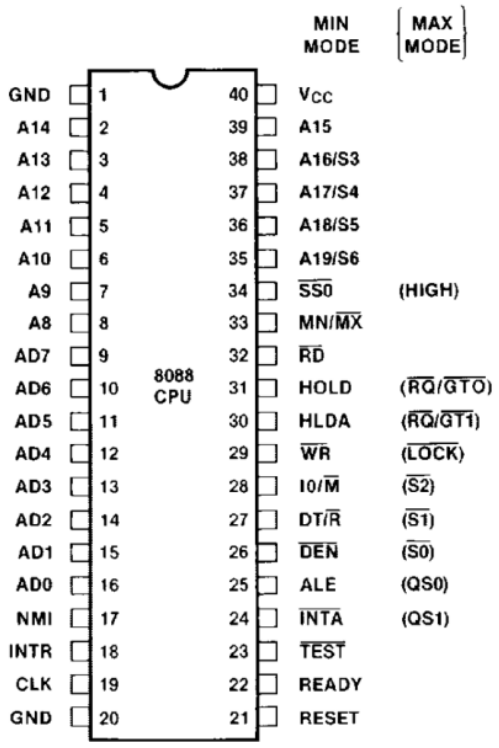

图 10: 8086 引脚图

8086 与 8088 对比：

1. 内部指令预期队列 6 字节 →4 字节
2. 8088：AD7 AD0 8086 AD15 AD0（速度快）
3. 8088 $S\bar{S}0$ 8086 $B\bar{H}E$/S7
4. 8088 I/O/$\bar{M}$ 8086 $\bar{M}$/I/O

引脚 $MN/\overline{MX} = 1$ 时，8086 工作在最小模式，微机中只有一个处理器。而当 $MN/\overline{MX} = 0$ 时，允许接入其他协处理器（运算处理器 8087 I/O 处理器 8089），此时系统总线有 8086 和总线控制器 8288 提供的信号共同形成，以支持多微处理器系统构成。

两种模式下的共用信号

1. $A_{16} \sim A_{19}/S_3 \sim S_6$（输出，三态）：高 4 位地址 $A_{16} \sim A_{19}$ 与状态 $S_3 \sim S_6$ 分时复用信号。状态 S_6 始终为低。S_5 表示中断允许标志，在每个时钟周期开始时被更新。S_4 和 S_3 用来指示 CPU 正在使用的段寄存器，其编码状态如下表。而高 4 位地址 $A_{16} \sim A_{19}$ 在 CPU 访问 I/O 时输出低电平，在一些特殊情况下还可以处于高阻状态。

S_4	S_3	指示的寄存器
0	0	ES
0	1	SS
1	0	CS 或不用
1	1	DS

2. $AD_0 \sim AD_{15}$（输入输出、三态）：地址 $A_0 \sim A_{15}$ 与数据 $D_0 \sim D_{15}$ 分时复用信号。CPU 读写主存或者 I/O 设备时，在总线周期的 T_1 时钟周期，地址信号有效，之后允许数据或状态信号有效。

3. $\overline{BHE}/S_7$（输入输出、三态）：高字节允许与状态 S_7 分时复用信号。在总线周期的 T_1 时钟周期 $\overline{BHE}$ 起作用。S_7 为备用状态。

4. RESET（输入）：复位信号，高电平有效，复位脉冲至少持续 4 个时钟周期。RESET 返回低电平时，CPU 重新启动。CPU 内部状态如下：

表2.5　复位后内部寄存器的状态

内部寄存器	内　容	内部寄存器	内　容
PSW	清除	SS	0000H
IP	0000H	ES	0000H
CS	FFFFH	指令队列缓冲器	清除
DS	0000H		

表 2.6　复位后各引脚的状态

引脚名	状　态	引脚名	状　态
AD_0～AD_{15}	浮动	$\overline{INTA}$	输出高电平后浮动
A_{16}/S_3～A_{19}/S_6	浮动	ALE	低电平
$\overline{BHE}/S_7$	浮动	HLDA	低电平
$\overline{DEN}$ ($\overline{S_0}$)	输出高电平后浮动	$\overline{RQ}/\overline{GT_0}$	高电平
DT/$\overline{R}$ ($\overline{S_1}$)	输出高电平后浮动	$\overline{RQ}/\overline{GT_1}$	高电平
M/$\overline{IO}$ ($\overline{S_2}$)	输出高电平后浮动	QS_0	低电平
$\overline{WR}$ ($\overline{LOCK}$)	输出高电平后浮动	QS_1	低电平
$\overline{RD}$	输出高电平后浮动		

5. READY（输入）：准备就绪信号，高电平有效。CPU 读写内存或 I/O 设备时，在总线周期 T_3 时钟周期采样 READY 信号。若为高，则表示设备已经准备好，若为低电平，则表示被访问的存储器或设备没有完成读写，需要插入等待周期 T_{WAIT}，在等待周期继续采样 READY 信号，直至 READY 变为有效（高电平），进入 T_4 周期，完成数据读写。

6. $\overline{TEST}$（输入）：测试信号，低电平，用于与其他处理器（8087）同步执行程序。CPU 执行 WAIT 指令时测试该引脚，当该信号无效时，CPU 进入等待状态（空转）；一旦该信号有效，CPU 退出等待，继续执行程序。这个信号在每个时钟周期的上升沿由内部电路进行同步。

7. INTR（输入）：可屏蔽中断请求信号，高电平有效。CPU 在每条指令执行的最后一个时钟周期采样该信号，以决定是否进入中断响应（INTA）周期。用软件复位中断允许标志（IF=0），可以屏蔽该中断请求。

8. NMI（输入）：非屏蔽中断请求信号，上升沿有效。该请求信号不可被屏蔽，中断一定会发生。NMI>INTR

9. $\overline{INTA}$（输出、三态）：对 INTR 中断请求的响应信号。在响应中断过

程中，由 $\overline{INTA}$ 送出两个负脉冲（两个 INTA 周期），在第二个 INTA 周期 CPU 获得外部中断源的中断向量码。最小模式由 8086 提供，最大模式时由 8288 提供。

10. $DT/\overline{R}$（输出、三态）：数据发送/接收控制信号，用于确定数据的传送方向。高电平控制数据发送，即 CPU 将数据写到主存或 I/O 接口；低电平控制数据接收，从主存或 I/O 接口读取数据。用于数据总线驱动器 8286/8287 或 74245 的方向控制。该信号在最小模式时用 8086 提供，在最大模式时用 8288 提供。

11. $\overline{DEN}$（输出、三态）：数据有效信号。该信号有效时，表示 $D_0 \sim D_{15}$ 上的数据有效。该信号在最小模式时由 8086 提供，最大模式时由 8288 提供。

仅在最小模式下使用的信号