TDA4 泊车应用中的超声波雷达集成方案

王力（Neo Wang） ***Central FAE***

摘要

Jacinto™ 7 TDA4 系列处理器是 TI 公司基于 Keystone 架构推出的最新一代汽车处理器，主要致力于辅助驾驶系统（ADAS）和自动驾驶（AD）领域芯片解决方案。TDA4 系列汽车处理器基于片上多核异构芯片架构，16nm 系统工艺不仅为芯片带来了高系统集成度，而且大幅降低了多功能高级汽车平台设计所需的外设复杂度以及成本。性能上，在提供高达 ASIL-D 的系统功能安全等级支持的同时，以领先于市场的性能/功耗比为深度学习/视觉加速提供了卓越的处理能力。

自动泊车 APA 以及代客泊车 AVP 作为目前 ADAS 市场中最为火热以及普遍的应用，其需要多个摄像头，超声波雷达传感器以及 ADAS 控制器的协同工作，从而实现对车位的寻找，识别以及泊入。在传统的 ADAS 控制器中，需要外置一个高功能安全等级的 MCU 对超声波雷达模块进行驱动、数据接收以及处理，并将处理结果传送至 SOC 端与摄像头数据做融合。但在 TI 最新 TDA4 处理器中，其内部集成了多颗高算力 MCU，能够用来直接驱动外部超声波雷达并获取障碍物数据，从而实现与摄像头数据的实时融合，提高了系统效率，同时大幅降低了硬件以及研发成本。这也是在 TI 汽车处理器中第一次直接集成超声波雷达模块。

本文通过多种方案实现了 TDA4 内部硬件模块对外部超声波雷达的驱动以及数据捕获，并通过实例测试，对各个方案进行了优劣对比，能够满足 TDA4 处理器在不同泊车应用场景下的超声波雷达集成需求。

目录

图

表

1. 系统介绍

在 ADAS 泊车应用中，需要调用安装在车辆四周的超声波雷达来实现对周围障碍物的扫描，并使用摄像头对静动态物体、行人、车辆以及车位等进行识别，然后通过对超声波雷达数据以及摄像头数据的实时融合，来获取可停车位位置以及停车位的长度，深度等信息，最后根据自身车辆与障碍物以及车位的相对关系，运用自身携带的泊车算法，来计算车辆入库的泊车轨迹，从而实现自动泊车。如图 1 所示，目前较为常规的是至少使用 12 路超声波雷达以及 4 路摄像头传感器来实现泊车功能。

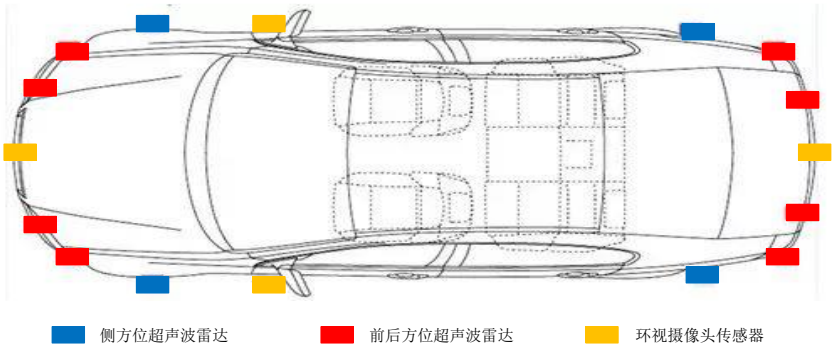


图 1 常见泊车应用中传感器接入方案

其中 4 个侧方位超声波雷达用于在泊车过程中检测车身周围的障碍物，从而避免剐蹭；8 个前后方位超声波雷达用于在泊车开始前进行车位的探测以及在泊车过程中提供前后障碍物信息；4 个环视鱼眼摄像头传感器用于在整个泊车应用中对周围环境以及物体的视觉探测和定位。摄像头传感器识别距离远，并能够识别色彩、形状以及方位等环境信息，但同时其容易受到光照烟雾等外界因素影响；超声波雷达不受外界光照烟雾等因素影响，但其仅能在一定范围内测量距离，却不能准确的获取物体方位。所以两者相辅相成，优势互补，信息融合，从而实现特定环境下的泊车应用。

1.1 TDA4 中的多核异构设计

以 TDA4VM 为例，其基于多核异构处理器架构，除了用于视觉加速，编解码以及深度学习的硬件核心外，不仅集成了两个高性能 ARM Cotex A72 核心，还分别在 MCU 域以及 MAIN 域分别集成了 2 个以及 4 个 ARM Cotex R5F 的实时处理器核心，如图 2 所示。

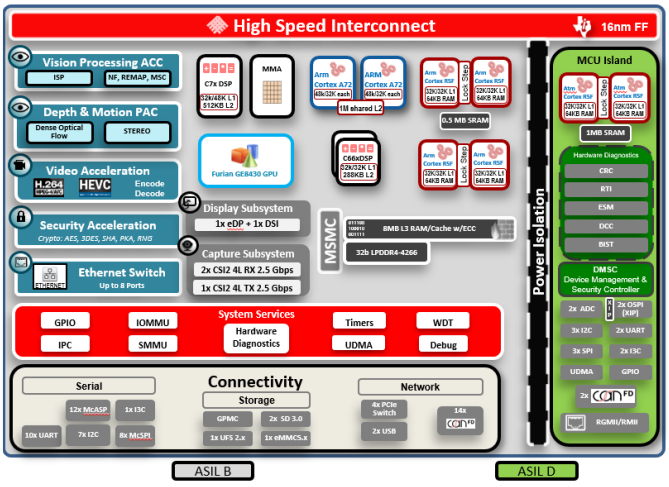


图 2 TDA4VM 系统结构框图

其中 A72 核心可用于运行上层操作系统，R5F 则用于处理对实时性以及延迟性要求较高的数据信息。在本文介绍的泊车应用中，可利用此 R5F 来对超声波雷达进行驱动以及信息处理，同时，由于 TDA4 中集成的 R5F 核心可运行在 1GHZ 频率下，故其能够满足在泊车应用下的超声波雷达速率要求。

1.2 USS 模块工作原理

超声波雷达 USS 模块的工作原理是通过超声波发送探头向外发送超声波，超声波在向外扩散过程中遇到障碍物就会产生反射波，然后通过接收探头对反射波进行接收，通过发送和接收到超声波的时间差来计算障碍物的距离，如图 3 所示。常用的探头工作频率有 40KHz，48KHz 以及 58KHz，一般来说频率越高，灵敏度越高，但是水平以及垂直方向的探测角度 FOV 就越小，所以一般就采用 40KHz 的探头。

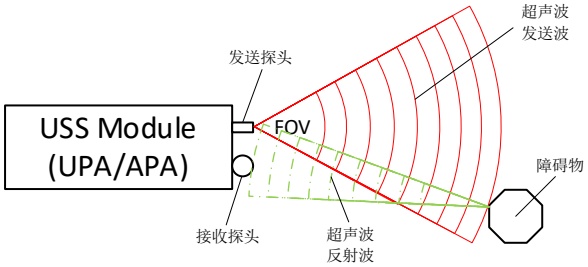


图 3 超声波雷达模块测障原理

超声波雷达不受烟雾影响，探测距离一般在 0.1~3 米之间，精度可以达到毫米级别。超声波的速度会收到温度的略微影响，其计算公式为:

$$V = V_0 + 0.607 * T$$

V_0 为温度 0℃时超声波传输速度，一般取 332 米/秒，T 为温度，单位℃。

如果能够记录到发送探头发送超声波时间为 t_s，接收探头接收到反射波时间为 t_r，两者时间差为 Δt，则可以得到障碍物与 USS 模块之间的距离 L 为：

$$L = \frac{\Delta t * V}{2} = \frac{\Delta t * (V_0 + 0.607 * T)}{2}$$

根据上述分析可以得知，其中发送超声波时间 t_s 以及捕获到反射波时间 t_r 的精确度将直接影响到超声波雷达对障碍物距离的计算。

1.3 传统泊车场景下的超声接入方案

在传统泊车场景下的超声波雷达接入方案中，需要外置一颗单独的 MCU 来发送 PWM 波给超声波雷达进行驱动，并使用这颗单独的 MCU 来对超声反射波进行接收并触发中断，在 MCU 内部对发送接收时间差进行计算并得到距离信息，最后将计算得到的距离信息通过 PCB 板载互联传输至 ADAS SOC 处理器中，ADAS 在得到距离信息之后在与视觉信息进行融合，处理以及决策，如图 4 所示。

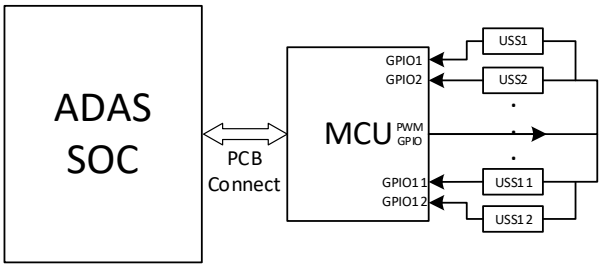


图 4 传统泊车场景下超声波雷达接入方案示意图

上述架构中传统的超声波雷达接入方案较为局限，因为不仅在成本上需要额外购置一颗频率以及算力充足的 MCU，而且在整个数据链路中，相比较于 USS 模块，ADAS SOC 得到的距离信息会有一定的延迟，这会给后面的多传感器数据融合带来时间同步等问题。

1.4 TDA4+USS 在泊车场景下的应用

在 TDA4 汽车处理器中，如 1.1 小节所述，其内部集成了多颗运行在 1GHz 频率的实时 MCU R5F，ARM Cotex R5F 的算力在 1GHz 下能达到 12KDMIPS，所以只需要调用 TDA4 中的某一颗 R5F，仅使用其部分算力就能够实现对 12 路超声波雷达模块的接入，如图 5 所示为 TDA4 在泊车场景下的超声波雷达接入示意图。

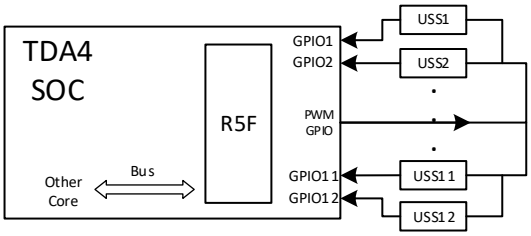


图 5 TDA4 在泊车场景下的超声接入方案示意图

同样的，其不仅在成本上能够得到大幅减少，由于其数据的捕获、计算、融合以及决策都在 TDA4 内部进行，所以其数据的实时性将得到大幅提高。

> 一般使用 MAIN 域的 R5F 用于超声波雷达模块的驱动以及数据处理，因为 MCU 域的 R5F 因为功能安全考虑，常用于处理高功能安全要求的任务。

另外的，TDA4 内部也集成了例如 EPWM，ECAP，GPTimer 等多个硬件模块，能够使用内部硬件模块对数据进行捕获以及处理，这样对处理器资源的负载也会大幅减小，并进一步保证了数据的实时性。此部分将会在后续章节展开介绍。

2. TDA4 驱动 USS 模块

一般的，超声波雷达需要外部输入脉冲宽度调制波 PWM 到模块中，作为激励脉冲驱动内部电路，从而使得超声波发送探头发送超声波信号。在 TDA4 处理器中，提供了两种高效且负载低的 PWM 产生方式，即使用 EPWM 硬件模块产生 PWM/HRPWM 以及使用 GPT 内部 timer 来产生 PWM。本节将基于软件 SDK8.0 以及 TDA4VM 开发板套件进行测试。

> HRPWM 是高分辨率 High Resolution 的缩写，其能够通过微边沿定位器实现在相同系统时钟的情况下把 PWM 的分辨率提高，从而应用在一些高分辨率 PWM 的场景中。一般情况下，再超声波雷达中，PWM 分辨率就能够满足系统需求，所以 PWM 和 HRPWM 的具体差别此处不进行赘述。

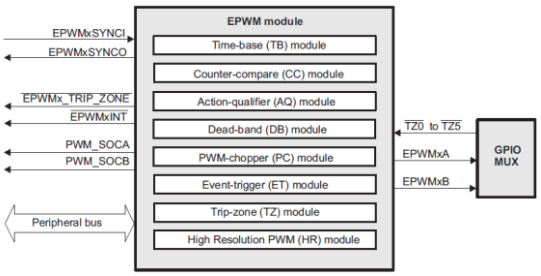
2.1. EPWM 硬件产生 PWM/HRPWM

在 TDA4VM 中，一共有 6 个名为 EPWM（Enhanced Pulse Width Modulation）的硬件模块，EPWM 可用于配置输出不同极性、占空比以及频率的 PWM 波。这 6 个 EPWM 硬件模块都在 MAIN 域，其中每个 EPWM 模块都有 EPWMxA 以及 EPWMxB 两个输出端口，所以理论上在 TDA4VM 中可利用 EPWM 模块一共产生 12 路极性占空比各不相同的 PWM 波输出，其中 HRPWM 只可在 EPWMxA 端口输出。每个 EPWM 模块的基地址如表 1 所示，本小节以 EPWM1 举例说明，其余端口可以此类推。

表 1 TDA4VM 中 EPWM 硬件模块基地址

Instance	Base Address
EHRPWM0_EPWM	0300 0000h
EHRPWM1_EPWM	0301 0000h
EHRPWM2_EPWM	0302 0000h
EHRPWM3_EPWM	0303 0000h
EHRPWM4_EPWM	0304 0000h
EHRPWM5_EPWM	0305 0000h

每个 EPWM 模块的内部构成以及对应的模块说明如图 6 所示，可以看到，每个 EPWM 模块都有独立的时钟基准以及计数比较模块，用户可以通过对每个 EPWM 模块的 TB 模块进行配置实现不同周期的 PWM 输出，或者通过配置 CC 模块来实现对 PWM 输出占空比的实时调整等。



Module	Comments
Time-Base (TB)	设置HSPCLKDIV以及CLKDIV等，根据系统时钟以及输出周期要求，计算出TB要设置的Ticks等
Counter-Compare (CC)	设置EPWMxA/B端口对应的计数值，从而控制PWM输出占空比以及高低电平切换时间等
Action-Qualifier (AQ)	当CC计数值溢出时采取的动作，例如拉高/拉低等
Dead-Band (DB)	PWM上升沿或下降沿延迟时间等
PWM-Chopper (PC)	控制产生PWM高频载波信号等
Event-Tigger (ET)	使能错误，事件触发中断等
Trip-Zone (TZ)	当有错误事件触发时采取的动作，例如拉全高，全低等
High-Resolution-PWM (HR)	使能高分辨率PWM输出模式

图 6 EPWM 硬件内部模块示意图

其中 TB、CC 以及 AQ 模块在设置 PWM 输出时控制了最基础的周期，占空比等设置。其中对于 TB 模块，其 Ticks 计算公式为：

$$PRD = \frac{\frac{SYS\_CLK}{HSPCLKDIV * CLKDIV}}{\frac{PWM\_FREQ}{2}} = \frac{2 * SYS\_CLK}{PWM\_FREQ * HSPCLKDIV * CLKDIV}$$

其中 *SYS_CLK* 为 125MHz，*HSPCLKDIV* 和 *CLKDIV* 分别为高速和常规下的 TB 模块分频指数，其取值以及对应分频倍数在寄存器 *EPWM_TBCTL* 中设置；*PWM_FREQ* 为预输出 PWM 的频率；*PRD* 为寄存器 EPWM_TBPRD 取值。由于 EPWM_TBPRD 寄存器为 16bit 寄存器，所以其最大取值为 65536；而 *HSPCLKDIV* 和 *CLKDIV* 预分频取值分别最大为 14 和 128；容易得出 EPWM 最大输出频率 *PWM_FREQ* 取整后约为 5Hz。

上述模块的执行顺序以及输出流程如图 7 所示，关于其参数配置参见 TRM 数据手册以及软件手册，在本测试示例中也会按照图 7 流程进行配置。

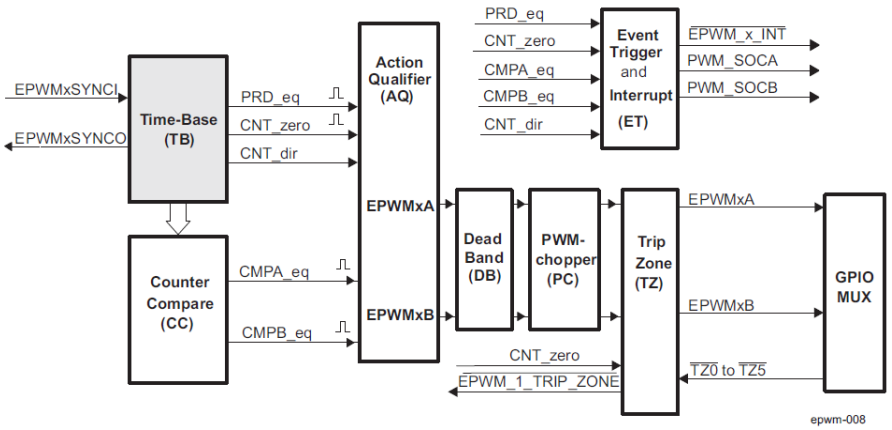


图 7 EPWM 内部模块配置流程示意图

在默认的 SDK8.0 中，有对应的 EPWM 基于 MCAL 的驱动，请按照下述流程进行编译并得到 bare mental 可执行文件。

```
$ cd mcusw/build
$ make pwm_app_epwm BOARD=j721e_evm SOC=j721e BUILD_PROFILE=debug CORE=mcu2_1 -j8
$ ls mcusw/binary/pwm_app_epwm/bin/j721e_evm/pwm_app_epwm_mcu2_1_debug.xer5f
```

默认测试程序将按照图 8 流程进行 PWM 输出，在此过程中，输出的 PWM 波极性为高，并对其占空比以及频率做实时调整，并通过 TDA4VM 的 U28，U29 两个 PIN 脚进行输出，对应为 GESI 板上 J22 测试管脚中的 EHRPWM1_A 以及 EHRPWM1_B 引脚进行波形输出。

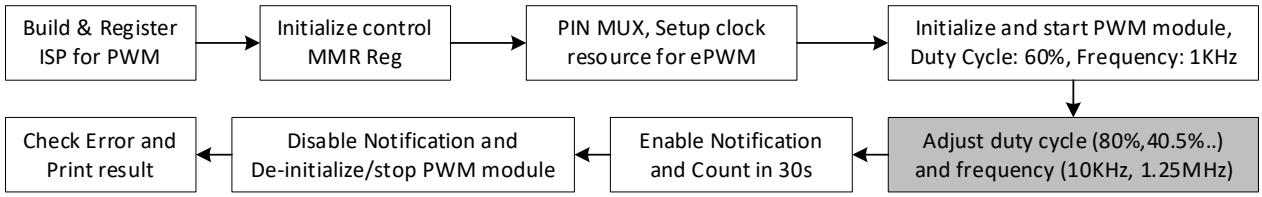


图 8 EPWM 输出 PWM 软件流程示意图

参照软件开发手册，将生成的可执行文件使用 CCS Load 的方式进行加载，TDA4 开发板需要设置为 No Boot 模式运行，其测试结果如第四章节所示。

2.2. GPTimer 硬件产生 PWM

类似的，TDA4 可以利用内部定时器 GPTimer 进行计数，并利用内部逻辑模块与预设值进行比较，从而控制特定引脚 TIMER_IO 进行 PWM 输出。

在 TDA4VM 中，共有 30 个硬件时钟模块，其中 MCU 域有 10 个，MCU_TIMER0 ~ MCU_TIMER9；MAIN 域有 20 个，TIMER0 ~ TIMER19。同时在 MCU 域中，有 10 个特定的 MCU_TIMER_IO 引脚可分别配置为 MCU 域 10 个定时器的输入或 PWM 输出，其中每个 MCU_TIMER_IO 引脚可配置链接到任意一个 MCU 域定时器 MCU_TIMERx 中；在 MAIN 域中，仅有 8 个 TIMER_IO 引脚可分别配置为 MAIN 域 20 个定时器的输入或 PWM 输出，其中每个 TIMER_IO 引脚可配置链接到任意一个 MAIN 域定时器 TIMERx 中。如图 9 所示为 TDA4VM 内部定时器和对应输出管脚模式控制器以及输出引脚的示意图。

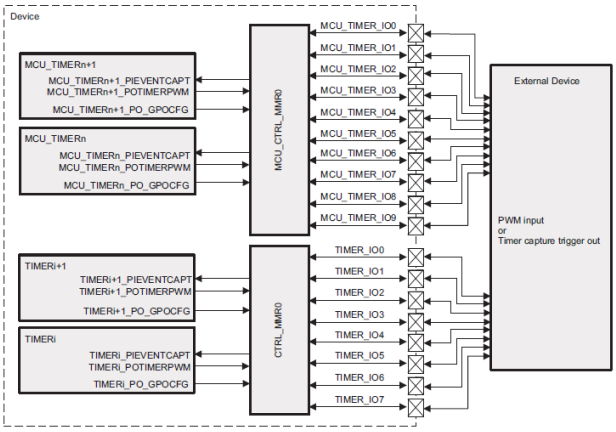


图 9 GPTimer 与 TimerIO 硬件映射图

在本文中，将基于 MAIN 域的 TIMER0 以及 TIMER_IO0 进行配置，并通过 TDA4VM 中 V6 管脚进行输出。值得注意的是，利用 GPTimer 硬件产生 PWM 输出，其主要寄存器配置流程如图 10 所示，详细寄存器配置请参照 TRM 数据手册。

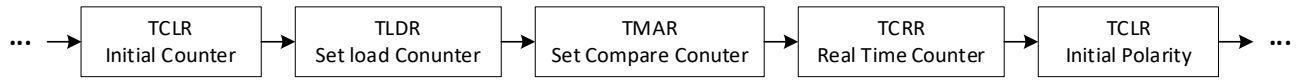


图 10 GPTimer 内部寄存器配置流程示意图

在默认的 SDK8.0 中，有对应的 GPTimer 基于 MCAL 的驱动，请按照下述流程进行编译并得到 bare mental 可执行文件。

```
$ cd mcusw/build
$ make pwm_app BOARD=j721e_evm SOC=j721e BUILD_PROFILE=debug CORE=mcu2_1 -j8
$ ls mcusw/binary/pwm_app/bin/j721e_evm/pwm_app_mcu2_1_debug.xer5f
```

默认测试程序将按照图 11 流程进行 PWM 输出，通过 TDA4VM 中的 V6 PIN 脚，复用为 TIMER_IO0 进行输出，在此过程中，输出的 PWM 波极性为高，并对其占空比以及周期做实时调整。

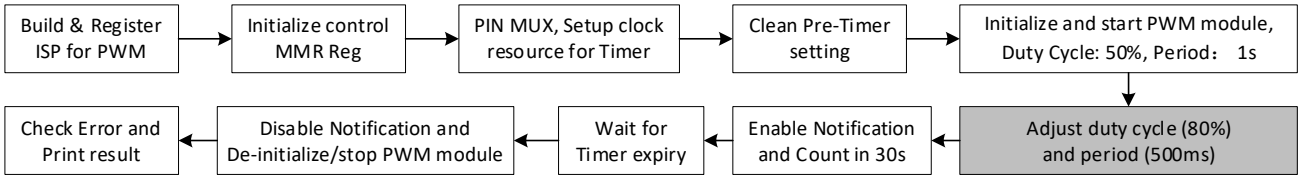


图 11 GPTimer IO 输出 PWM 软件配置流程图

GPTimer 输出 PWM 程序测试方法与 2.1 小节一致，其测试结果如第四章节所示。

3. TDA4 捕获 USS 回波

TDA4 按照超声波雷达模块要求输出 PWM 波之后，超声波雷达 USS 模块会按照 1.2 小节所述工作并返回超声反射回波，此时需要 TDA4 实时捕获超声反射回波并获取当前定时器数值，通过定时器中的时间差来计算超声波发送接收时间差，从而在 TDA4 R5F 核心中计算超声波雷达测绘距离。

传统低端处理器会采用 GPIO 加中断触发的方式进行回波捕获，这种方式极其占用处理器资源且捕获准确度低，已经被市场淘汰，故在本文中不进行讨论。

在 TDA4 处理器中提供了两种方法进行高效准确的 USS 回波捕获，均使用硬件进行触发、捕获以及搬运，本章将基于 ECAP 硬件模块以及 GPIO+UDMA 的方式分别进行介绍。

3.1. ECAP 硬件捕获

在 TDA4VM 中，一共有三个 ECAP 模块位于 MAIN 域，其分别对外引出 PIN 脚 ECAP0_IN_APWM_OUT ~ ECAP2_IN_APWM_OUT，每个 ECAP 都可以分时复用为捕获模式或 PWM 输出模式。当 ECAP 配置为捕获模式时，对应管脚配置为输入引脚，其中每个 ECAP 可以用于捕获外部 PWM 输入信号，例如边沿检测以及边沿计数，并通过当前触发来获取计算得到当下时间戳，占空比以及周期等信息。同样的，ECAP 模块由于内部集成有 PWM 逻辑单元，也可配置为 PWM 输出模式，对外输出指定占空比以及周期的 PWM 波。其大致结构示意图如图 12 所示。

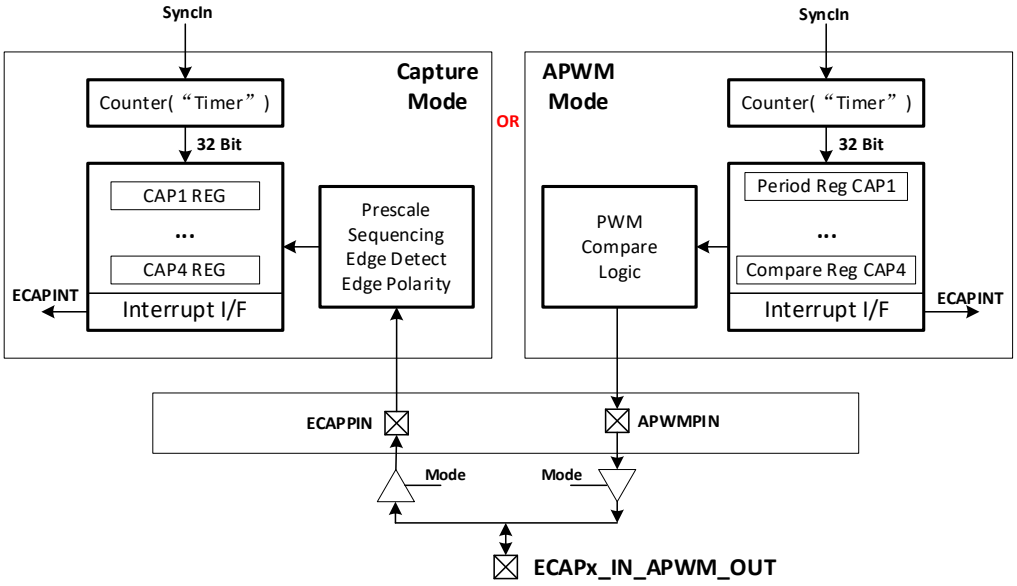


图 12 ECAP 结构示意图

一个 PIN 脚在某一时刻只能配置为 ECAP 输入或 APWM 输出功能。在 SDK8.0 中，有基于 MCAL 的 ECAP 软件驱动以及示例程序 ICU，但 ECAP 的驱动以及 ICU 测试程序仅实现了 ECAP 硬件模块的部分功能，例如预分频等驱动将在后续加入。可按照下述流程进行编译并得到 ICU bare mental 可执行文件。

```
$ cd mcusw/build
$ make icu_app BOARD=j721e_evm SOC=j721e BUILD_PROFILE=debug CORE=mcu2_1
$ ls mcusw/binary/icu_app/bin/j721e_evm/icu_app_mcu2_1_debug.xer5f
```

在 ICU 测试程序中进行了信号测量、时间戳获取、跳变沿检测以及跳变沿统计四部分功能测试，其中信号测量包括占空比、高/低电平时间以及周期等。其软件流水示意图如图 13 所示。

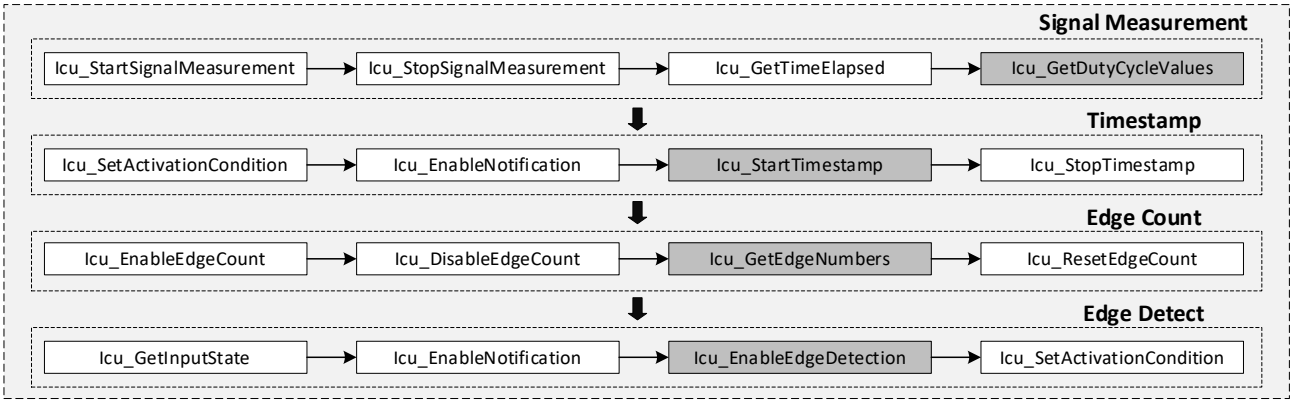


图 13 ECAP 捕获 PWM 信号软件流程图

用户可基于 ECAP 模块实现对外部 PWM 波形的捕获，并实时采集时间戳信息，从而能够得到发射波与反射波之间的时间差，进而估算出障碍物距离。本示例基于管脚 EHRPWM1_A 以及 EHRPWM1_B 管脚进行 PWM 输出，进而模拟超声波雷达反射波，对应 TDA4VM 中的 U28，U29 管脚；并基于 ECAP1_IN_APWM_OUT 管脚进行捕获，对应 TDA4VM 中的 V6 管脚。

ECAP 测试程序 ICU 测试方法与 2.1 小节一致，测试时需要将 EHRPWM1_x 管脚与 ECAP1_IN_APWM_OUT 管脚进行连接，其测试结果如第四章节所示。

3.2. GPIO+UDMA 硬件捕获

上述使用 ECAP 捕获外部 PWM 回波是基于 TDA4 内部硬件模块实现，其相比较于 GPIO+软中断，能够显著的降低硬件 loading，但是很明显的是，其受到 ECAP 硬件模块个数的制约，能够支持的捕获路数有限。本小节将介绍基于普通的 GPIO 口进行外部 PWM 的捕获，并触发 UDMA 搬运当前定时器的数值，并且在整个过程中使用 UDMA 对时间数据进行搬运，能够实现突破硬件模块限制的前提下，显著的降低对处理器资源的占用。

UDMA 全称 Unified Direct Memory Access，是 TI 基于 TDA4 系列处理器设计的新一代 DMA 通道控制引擎，能够同时支持最多 512 路数据包，直接从内存映射数据结构在系统中的 UTC 数据通道，反之亦可，UDMA 的详细描述请参照 TRM 数据手册。其中 UDMA 中数据的传输依赖于名为 TRPD（Transfer Request Package Descriptor）的数据结构，其中定义了包括 UDMA 搬运数据源地址 ADDR，目的地址 DADDR 等配置信息，其数据结构如图 14 所示。在本示例中，需要将源地址设置为 GTC Timer 计数寄存器基地址（GTC_CNTCV_LO – 0x00A9 0008h），目的地址为用户定义用来储存事件信息的结构体 buffer 中，其余字节参照示例代码以及 TRM 数据手册。

word 15		word 14		word 13		word 12
DICNT3	DICNT2	DICNT1	DICNT0	DDIM3		DDIM2
word 11		word 10		word 9		word 8
DADDR				DDIM1		FMTFLAGS
word 7		word 6		word 5		word 4
DIM3		DIM2		ICNT3	ICNT2	DIM1
word3		word 2		word 1		word0
ADDR				ICNT1	ICNT0	FLAGS

图 14 UDMA TRPD 结构

本小节将基于 SDK7.1 进行测试，其完整的补丁包详见 E2E 所示，并为了测试方便，在本测试用例中，使用软件对 GPIO 进行定时翻转，进而模拟外部 PWM 信号的输入，然后将对应的 Timer 数值存储在指定的缓存 buffer 中，在 CCS 中将 buffer 进行输出，从而验证触发以及采集，传输的正确性。整个示例程序的搭建流程如下所示。

1. 基于 SDK7.1 制作可用于启动 RTOS 系统的 SD 卡，详细步骤参照软件使用手册。
2. 按照下述流程安装补丁包，并编译得到 Vision APPs 测试用例程序，补丁包附在 E2E 链接中。

```
$ cd vision_apps
$ git am 0001-reproduce-the-SV-DMA-issue-in-EVM-and-solved-it.-but.patch
$ make -s -j8 sdk
$ make -s -j8 vision_apps
```

3. 上述补丁及编译指令将生成 GPIO+UDMA 测试用例并打包在 MCU2_0 系统镜像中，其路径位于:

   ~/ti-processor-sdk-rtos-j721e-evm-07_01_00_11/vision_apps/out/J7/R5F/SYSBIOS/debug/vx_app_tirtos_linux_mcu2_0.out

在 PC 机中插入 SD 卡，并使用下述命令将编译得到的库文件以及镜像文件更新到 SD 卡中。

```
$ make linux_fs_install_sd
```

4. 由于在 SDK7.1 中，默认的 TIFS 有基于 firewall 的限制，其关于在 DMSC 中进行的 UDMA 中断注册受到限制，所以为了快速 setup，可使用 SDK7.1 之后 SDK 版本的 TIFS 源文件，并将其在 SDK.1 中进行编译，从而得到能够在 SDK7.1 中支持 GPIO+UDMA 的 TIFS 文件。本小节使用 SDK7.2 作为示例，参照下述步骤进行编译。理论上，在 SDK7.1 后续 SDK 进行此示例移植时，可省去 TIFS 更新这一步。

```
$ cp ~/ti-processor-sdk-rtos-j721e-evm-07_03_00_07/pdk_jacinto_07_03_00_29/packages/ti/drv/sciclient/soc/sysfw/binaries/ti-fs-firmware-j721e-gp.bin ~/ti-processor-sdk-linux-j7-evm-07_01_00_10/board-support/prebuilt-images
$ cd ~/ti-processor-sdk-linux-j7-evm-07_01_00_10
$ make sysfw-image
$ cp ~/ti-processor-sdk-linux-j7-evm-07_01_00_10/board-support/k3-image-gen-2020.08b/sysfw-j721e-evm.itb /media/wangli/BOOT/sysfw.itb
```

5. 至此，所有 SD 卡启动文件均以更新，需要在 CCS 中对数据进行捕获并分析，从而对采集的数据正确性进行验证，其验证方法及结果参见 4.2 小节。

其软件流水示意图如图 15 所示，由图可知，首先需要对 UDMA 模块以及通道初始化完成，其中包括 TRPD 的格式以及事件类型等；在实际传输过程中，需要将搬运的值以及触发函数集成到 Event Callback 函数中；待所有传输完成后，对资源进行注销。

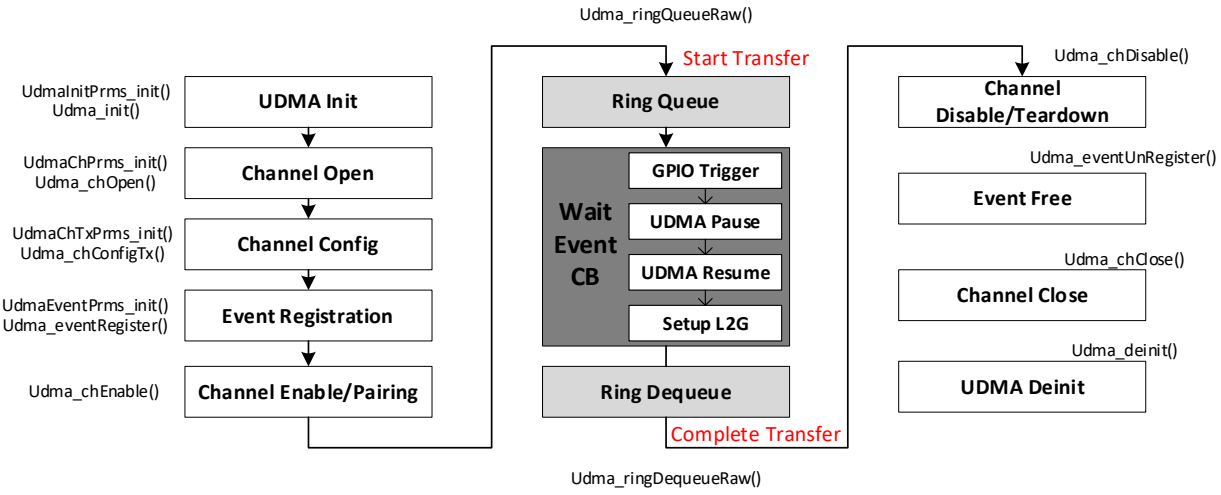


图 15 GPIO+UDMA 捕获 PWM 信号软件流程图

值得一提的是，在本示例代码中，考虑到实际超声波雷达应用中，存在数据采集暂停以及恢复的应用场景，设计并验证了在软件控制数据采集暂停/恢复的代码实现。其功能验证详见 4.2 小节。

4. 测试结果验证

4.1. TDA4 PWM 输出测试

TDA4 EPWM 硬件模块产生 PWM 输出如图 16 所示，可以看出输出 PWM 波形的周期以及占空比等参数都输出正常。

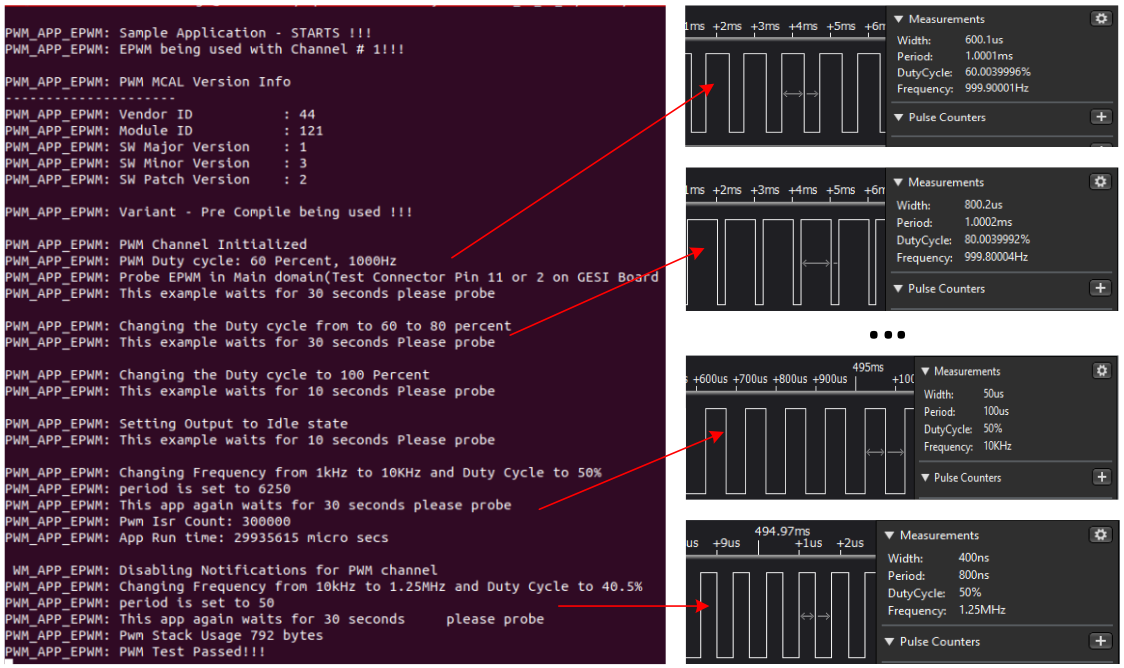


图 16 EPWM 硬件输出 PWM

TDA4 GPT Timer 硬件模块基于 TimerIO 产生 PWM 输出如图 17 所示，其输出 PWM 波形的周期以及占空比输出正常。

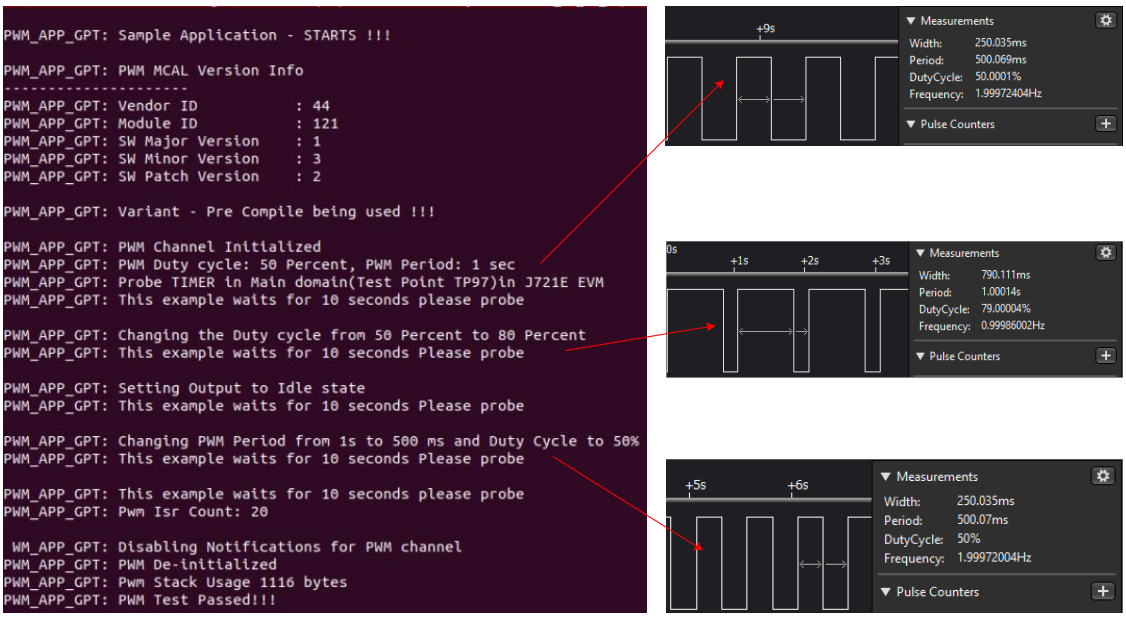


图 17 GPT Timer 硬件模块基于 TimerIO 输出 PWM

4.2. TDA4 USS 回波捕获测试

ECAP 捕获超声波回波测试如图 18 所示，可看出对于周期为 1KHz 以及 2.5MHz 的 PWM 波，分辨检测出的单边缘跳变为 1000 次以及 25000 次，双边缘跳变为 2000 以及 50000 次，检测结果与理论保持一致。

```
ICU_APP: ICU MCAL Version Info
---------------------
ICU_APP: Vendor ID          : 44
ICU_APP: Module ID          : 122
ICU_APP: SW Major Version   : 1
ICU_APP: SW Minor Version   : 3
ICU_APP: SW Patch Version   : 2

ICU_APP: Variant - Pre Compile being used !!!

ICU_APP: EPWM Channel Initialized
ICU_APP: EPWM Duty cycle: 60 Percent, 1000Hz
ICU_APP: Use EPWM (Pin 2 in J22 on GESI Board) as input to ECAP test point (TP30 on SOM
ICU_APP: Edge Detect Mode!
ICU_APP: input state is ICU_IDLE
ICU_APP: SignalNotification for Double Edge Detection Reached in 1 sec: 2000
ICU_APP: SignalNotification for Single Edge Detection Reached in 1 sec: 1000
ICU_APP: input state is ICU_ACTIVE

ICU_APP: Changing EPWM Frequency from 1000Hz to 25000Hz and Duty Cycle to 50%
ICU_APP: Edge Detect Mode!
ICU_APP: input state is ICU_IDLE
ICU_APP: SignalNotification for Single Edge Detection Reached in 1 sec: 25000
ICU_APP: input state is ICU_ACTIVE
ICU_APP: SignalNotification for after disabling notification Edge Detection Reached: 0
ICU_APP: Calling Deinit
ICU_APP: Icu Stack Usage 792 bytes
ICU_APP: ICU Test Passed!!!
```

图 18 ECAP 捕获超声波回波

GPIO+DMA 捕获超声波回波测试如图 19 所示，其中 0xd2400d00 为 UDMA 搬运目的 appChObj->destBuf 数组首地址，本验证方法在 App_GpioDmaTest 函数中添加断点并将 appChObj->destBuf 数组在 CCS 中另存为，能够得到每次 GPIO 触发 UDMA 搬运 GTC 定时器的值。

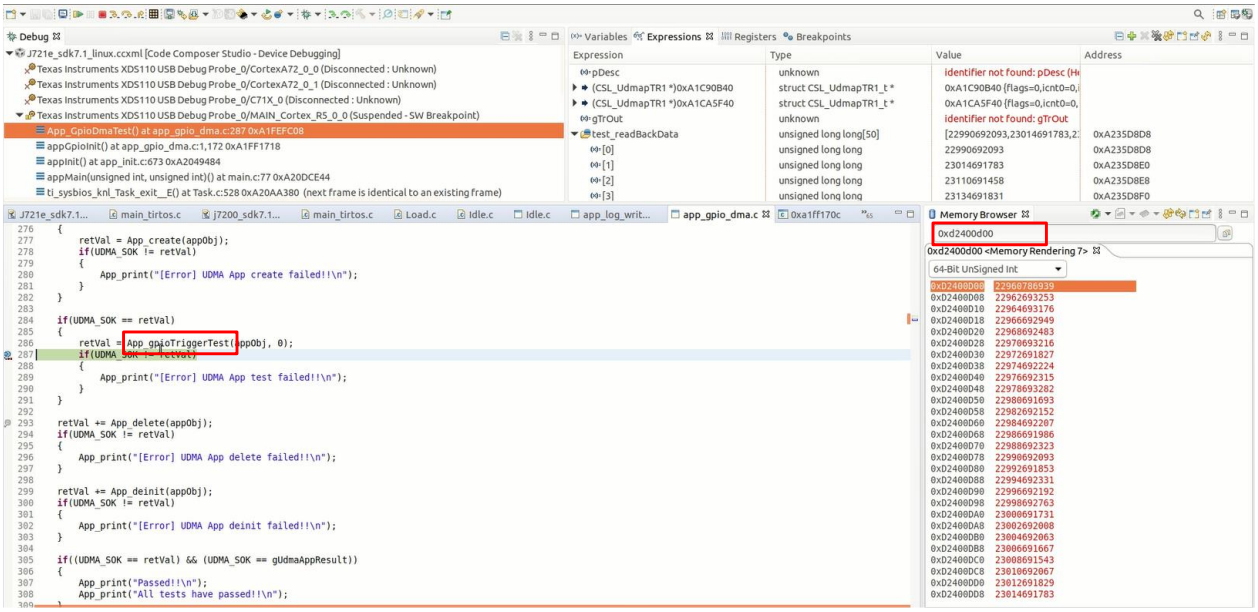
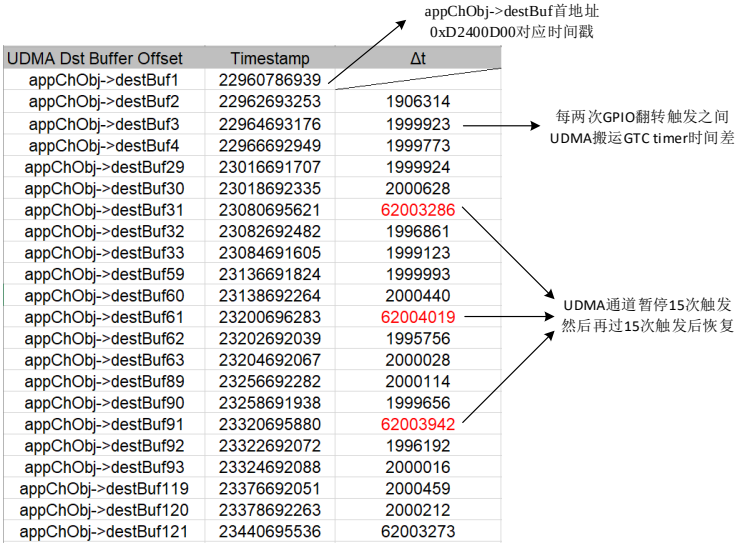


图 19 GPIO+DMA 捕获超声波回波

其数据整理后如图 20 所示，如图 20 中 A 列所示为对应数组中的不同偏移量，其中初始地址 0xD2400D00 所对应的数组取值对应图 20 中 A 列 appChObj->destBuf1 偏移所示；对于每一次 GPIO 电平翻转，其对应获取的时间戳如图 20 中 B 列所示；每两次触发之前的时间差如图 20 中 C 列所示。



UDMA Dst Buffer Offset	Timestamp	Δt
appChObj->destBuf1	22960786939	
appChObj->destBuf2	22962693253	1906314
appChObj->destBuf3	22964693176	1999923
appChObj->destBuf4	22966692949	1999773
appChObj->destBuf29	23016691707	1999924
appChObj->destBuf30	23018692335	2000628
appChObj->destBuf31	23080695621	62003286
appChObj->destBuf32	23082692482	1996861
appChObj->destBuf33	23084691605	1999123
appChObj->destBuf59	23136691824	1999993
appChObj->destBuf60	23138692264	2000440
appChObj->destBuf61	23200696283	62004019
appChObj->destBuf62	23202692039	1995756
appChObj->destBuf63	23204692067	2000028
appChObj->destBuf89	23256692282	2000114
appChObj->destBuf90	23258691938	1999656
appChObj->destBuf91	23320695880	62003942
appChObj->destBuf92	23322692072	1996192
appChObj->destBuf93	23324692088	2000016
appChObj->destBuf119	23376692051	2000459
appChObj->destBuf120	23378692263	2000212
appChObj->destBuf121	23440695536	62003273

图 20 GPIO+DMA 捕获超声波回波数据分析

为了展示方便，图 20 中对中间部分数值进行了折叠，但是可以看出，每两次 GPIO 翻转所捕获的 Timer 数值为 200ms 左右，此结果与 App_triggerGpio 中的设置保持一致。其中每 30 次触发进行一次 UDMA 暂停/恢复，所以在偏移量为 31，61，91 时时间差约为 6200ms，约等于 31 次 GPIO 翻转时间差，与理论计算结果保持一致。

5. 方案对比分析及总结

在上述章节中，对 TDA4 处理器中接入超声波雷达的方案进行了剖析，并针对 TDA4 输出 PWM 驱动超声波雷达，以及 TDA4 捕获超声反射波信号两个链路分别进行多方案实现。EPWM 以及 GPTimer 可用于输出多路自定义占空比以及周期的 PWM 波，ECAP 以及 GPIO+UDMA 可用于捕获多路超声回波并记录当前时间戳，从而计算出障碍物距离。相比较于纯软件，所有方案使用 TDA4 内部硬件模块进行实现，能够大幅降低对系统资源的占用。

5.1. 方案对比分析

如表 2 所示，针对 EPWM 和 GPTimer 输出 PWM，以及 ECAP 和 GPIO+UDMA 捕获超声回波进行了优劣分析。由于在泊车应用中，除了超声波雷达，往往也会接入多种外设，所以不同系统的外设种类，数量以及 PIN 脚复用各不相同。用户可根据不同的系统设计以及应用场景，选择 TDA4 在泊车应用中的合适的超声波雷达集成方案。

表 2 TDA4 超声波雷达驱动方案对比

	TDA4 输出 PWM		TDA4 捕获 PWM	
实现方案	EPWM	GPTimer IO	ECAP	GPIO+DMA
优势性	a. 可输出至多 12 路不同周期以及占空比的 PWM b. 支持输出至多 6 路 HRPWM	a. 可输出至多 18 路 PWM b. 20 路在主域，10 路在 MCU 域，输出配置灵活	驱动简便，处理器资源占用低	a. 输入捕获路数理论上至多支持 30 路独立输入。 b. UDMA 搬运数据，处理器资源占用低
局限性	a. 输出 PWM 不能超过 12 路 b. 所有 EPWM 在主域，不支持 MCU 域输出	不支持输出 HRPWM	捕获路数受限，最多仅支持 3 路输入	驱动较为复杂

5.2. 总结

在传统泊车应用中，处理器需要外接一颗 MCU 用作超声波雷达的驱动以及数据处理，这样不仅增加了成本，并且数据传输以及处理效率都不高。在 TI 最新一代处理器 TDA4 中，集成了多颗实时处理器核心 R5F，同时内部还集成了多个可产生 PWM 的硬件模块 EPWM 和 GPTimer IO，以及可实现捕获外部 PWM 信号的 ECAP 以及

UDMA 等硬件模块，可实现在不占用处理器资源的前提下，高效完成对超声波雷达的驱动。本文通过多种方案实现了对超声波雷达的集成，并对其优势以及局限性进行了逐一分析，用户可根据不同系统设计需求自行选择。

6. 参考文献

1. TDA4VM Jacinto™ Processors for ADAS and Autonomous Vehicles Silicon Revisions 1.0 and 1.1 datasheet (Rev. J)
2. DRA829/TDA4VM/AM752x Technical Reference Manual (Rev. B)
3. TDA4 SDK8.0 MCUSW Component User Guide

重要声明和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265