TriCore™ AURIX™ 家族系列

32 位

多脉冲产生驱动程序

AP32226

应用笔记

V1.0 2014-05

微控器

免责条款

为方便客户浏览，英飞凌以下所提供的将是有关英飞凌产品及服务资料的中文翻译版本。该中文翻译版本仅供参考，并不可作为任何论点之依据。虽然我们尽力提供与英文版本含义一样清楚的中文翻译版本，但因语言翻译和转换过程中的差异，可能存在不尽相同之处。因此，我们同时提供该中文翻译版本的英文版本供您阅读，请参见 www.infineon-ecosystem.org。并且，我们在此提醒客户，针对同样的英飞凌产品及服务，我们提供更加丰富和详细的英文资料可供客户参考使用。请详见 www.infineon.com

客户理解并且同意，英飞凌毋须为任何人士由于其在翻译原来的英文版本成为该等中文翻译版本的过程中可能存在的任何不完整或者不准确而产生的全部或者部分、任何直接或者间接损失或损害负责。英飞凌对于中文翻译版本之完整与正确性不担负任何责任。英文版本与中文翻译版本之间若有任何歧异，以英文版本为准，且仅认可英文版本为正式文件。

您如果使用以下提供的资料，则说明您同意并将遵循上述说明。如果您不同意上述说明，请不要使用本资料。

版本 2014/05

出版发行：
英飞凌科技公司
上海，中国

免责条款

本应用笔记中给出的信息仅作为实现英飞凌器件的建议，不得被视为英飞凌器件的任何特定功能、条件或质量作出的任何说明或保证。此应用笔记的接受者必须在实际应用中判定此种描述的任何功能。英飞凌科技在此否认承担此应用笔记中任何和所有信息相关的任何形式的保证和责任（包括但不限于不侵犯第三方知识产权）。

信息

有关技术、交货条款及条件和价格，请与您最近的 Infineon Technologies 办事处联系。

警告

由于技术要求，组件可能含有危险物质。如需相关型号的信息，请与您最近的 英飞凌科技办事处联系。如果可能合理地预期此类组件的故障会导致生命支持器件或系统发生故障或影响该器件或系统的安全性或有效性，则英飞凌提供的组件仅可用于获得英飞凌明确书面批准的生命支持器件或系统。生命支持器件或系统的目的是植入人体或支持和/或保持并维持和/或保护生命。如果出现故障，则可能危及使用者或他人的人身安全。

文献修订史			
日期	版次	修订人	修订内容
October 2012	V1.0	Alfredo Baratta	出版

商标：

TriCore™ and AURIX™ are registered trademarks of Infineon Technologies Ltd.

请留下您的宝贵建议
您是否认为本文档中的任何信息存在错误，含糊不清或遗漏？您的宝贵意见和建议将帮助我们持续不断地改进文档质量。请将您的建议（请注明文档的索引号）发送电子邮件至：ctdd@infineon.com

目录

1 导言

汽车动力总成系统的应用要求产生不同种类的脉冲,以控制喷油器,火花塞和其它种类的传动装置。

英飞凌 AURIX™家族产品中引进的通用定时器模块（GTM)提供了一个完美的解决方案，在整个汽车电子领域满足产生不同脉冲的要求。

1.1 对读者说

请注意该文档面向在发动机位置管理系统中已有一定经验的读者。

1.2 范围和目的

该文档描述了顶层设计，和多功能脉冲发生器（MPG)驱动程序设计所需的思想。

该文档还包含了主要的架构设计决策，并为开发组所有的成员提供了一个涉及设计，实现，验证和集成活动的通用框架。

此外，在产品生命周期的部署和维护阶段，该文件还是知识转移的材料。

1.3 参考文献

[1] 应用笔记 AP32212, GTM Engine Position Driver

[2] AURIX™ User Manual

[3] GTM 1.4.2 Errata

2 多脉冲产生驱动程序（MPG）

MPG 驱动程序必须能够产生不同种类的脉冲以满足最普通的汽车应用需求。

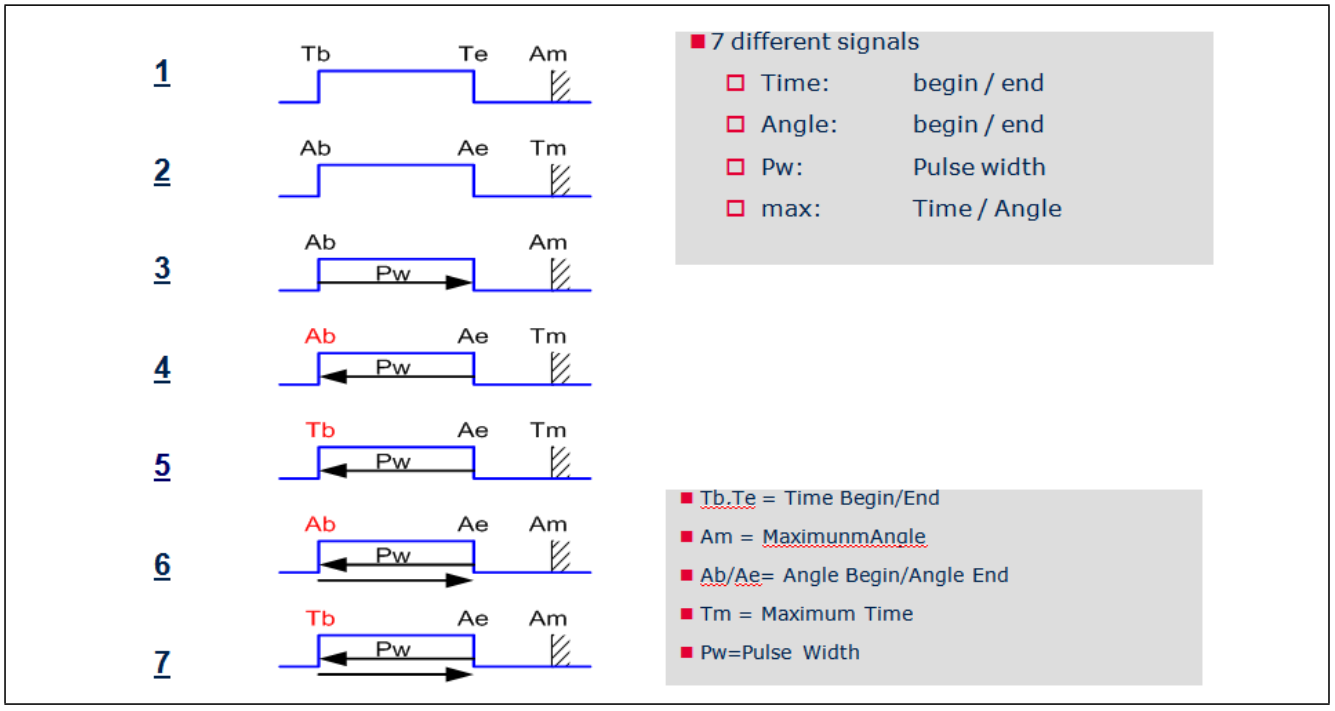

图 1 动力总成里的普通脉冲

接下来的章节详细地描述了每种脉冲。脉冲可被分为以下几种类型：

- 角度-角度脉冲
  - 脉冲以一定的角度开始并结束，使用角度计数器的绝对值。
  - 该脉冲类型被优先用于产生点火脉冲。
- 角度-时间脉冲
  - 该脉冲以一定的角度（绝对值）开始，并在一段定义的时间之后（绝对脉冲长度）结束。
  - 和直接喷油脉冲一样，该脉冲类型用于产生时间优先的点火脉冲。
- 时间-时间脉冲
  - 脉冲在一定的时间内开始并结束，该时间指计时器的绝对值。
  - 该脉冲类型可在固定的时间模式里被用于产生点火脉冲。
- 时间-角度脉冲
  - 终止角是一个绝对值，而开始角度的计算在脉冲长度内参照终止角。
  - 该脉冲类型被用于产生正常的点火脉冲。

2.1 系统架构

对于 MPG 驱动程序的实现,通用定时器模块 GTM 扮演着至关重要的角色。尤其是对于 ATOM(连接先进路由单元的定时器输出模块）这个模块，它需要与多个其它 GTM 子模块协同运行，以便能够实现一个较低占用 CPU 但仍能管理并产生不同的脉冲的系统。

接下来将对整个系统和所涉及的所有子模块进行基本的描述。

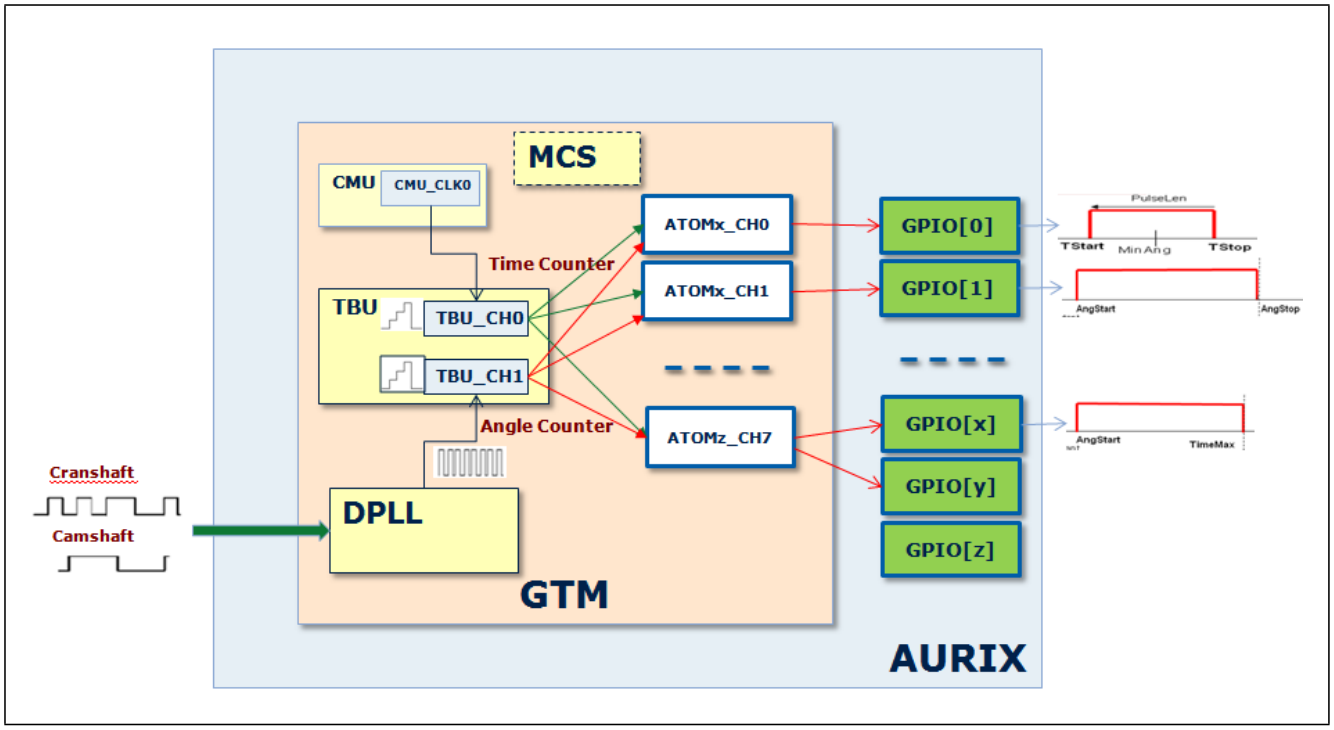

图 2 MPG 驱动程序硬件系统架构

在此需要明白的一个重要概念是 MPG 驱动程序无法作为一个独立的驱动程序工作。它还需要发动机位置（EP)驱动程序管理曲轴和/或凸轮轴信号，并使用 GTM DPLL 扩展角度基准。

注释：有关 EP 驱动程序和 DPLL 的描述，请参阅应用笔记 AP32212，“使用 GTM 的发动机位置驱动程序”。

2.2 ATOM(连接先进路由单元(ARU)的定时器输出模块)

由于 ATOM 自身就连接了 ARU 单元,因此无需 CPU 的介入,它就能产生复杂的输出信号。

每个 ATOM 子模块包含了 8 个输出通道，这 8 个输出通道在多个可配置操作模式下相互独立工作。

下图是 ATOM 子模块框图：

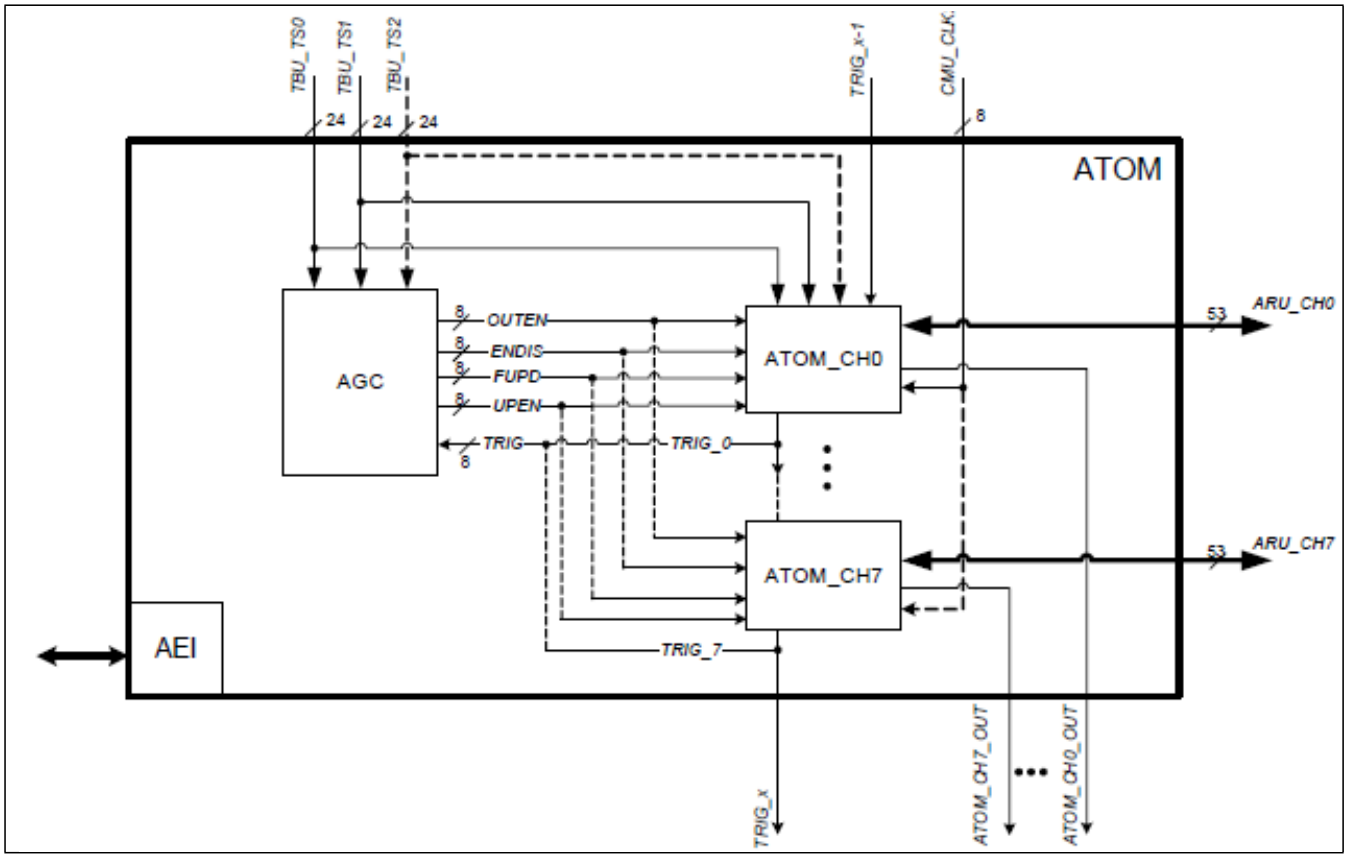

图 3 ATOM 框图

每个 ATOM 通道可在 4 种不同的信号输出模式下操作：

•ATOM 信号立即输出模式（SOMI）
•ATOM 信号比较输出模式（SOMC）
•ATOM 信号 PWM 输出模式（SOMP）
•ATOM 信号串行输出模式（SOMS）

本文件主要专注于 SOMC 模式。

每个 ATOM 通道都有两个捕获和比较单元(CCU0 和 CCU1)。

在 SOMC 模式下，可以使用这两个单元（需要的话也可以组合使用），以在发生匹配事件时引发中断和/或触发另外一个比较单元。

一般来说，CCU0 用于执行基于时间的比较，CCU1 用于基于位置/角度的比较。

以下是 ATOM 最重要的一些特性:

•每条通道的专用 PWM 时基计数器（24 位宽）。
•给两个比较寄存器的专用影子寄存器以及通道控制位的影子寄存器。
•时钟预分频器的变化考虑到了具有不同分辨率系数的宽范围 PWM 周期。
•全局触发单元以启/停，启用和停用多个并行的 ATOM 通道的选择。
•有符号的大于/等于比较器，用于检测时基溢出信号。
•ARU 连接，用于自动重载新数据值并控制信号，而无需 CPU 介入。

下图是典型的 ATOM 用途示例:

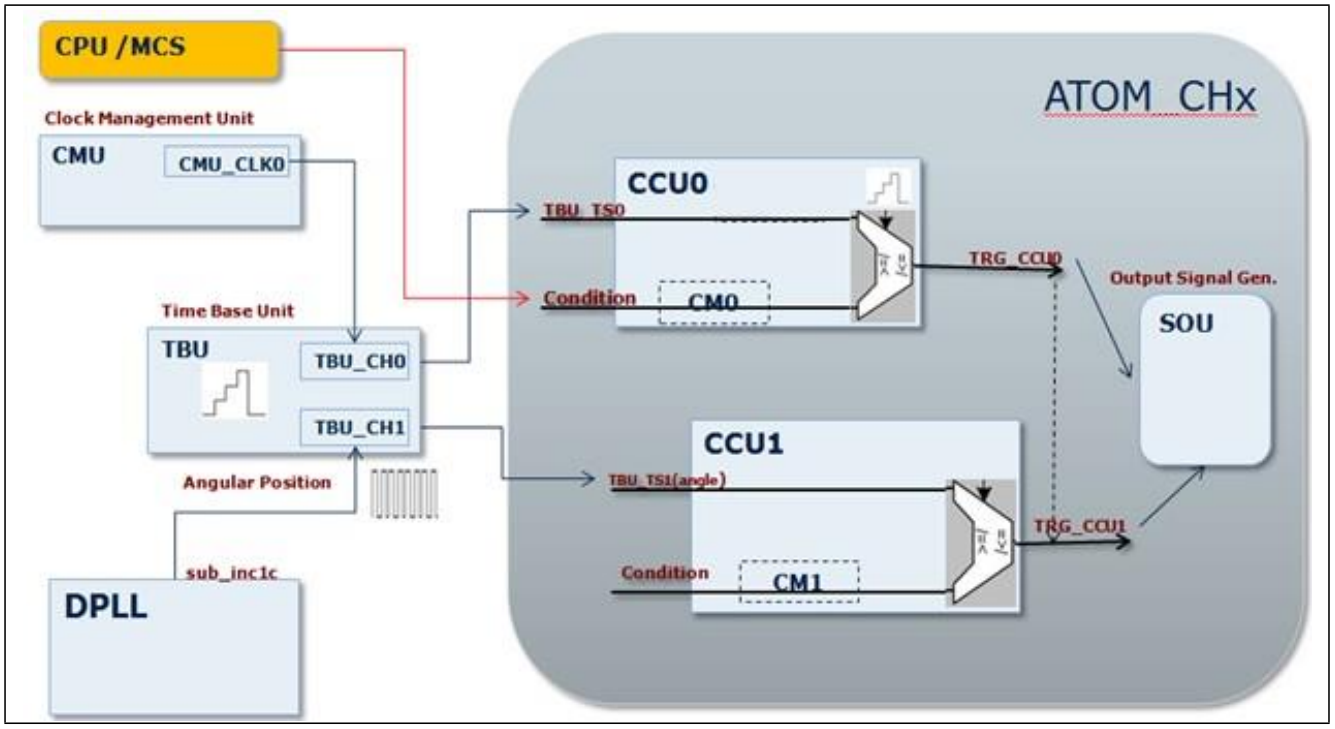

图 4 ATOM SOMC 示例

在 SOMC 模式下，两个 CC 单元，即 CCU0 和 CCU1 的表现取决于不同的情况，比如使用 ARU 控制 ATOM。在特殊情况下，这两个单元的表现受到下列事件的控制:

- ARU 停用事件
  - ACB 位的 bit 4 到 bit 2(ATOM[i]_CH[x]_CTRL 寄存器)
- ARU 使能事件
  - ACBI 位字段(ATOM[i]_CH[x]_STAT)，和通过 ARU 控制位的第 52 位到 48 位而被更新的 ACB 位字段。

根据 SL 位里预定义的信号电平，并结合可在 ATOM[i]_CH[x]_CTRL 或 ATOM[i]_CH[x]_STAT 寄存器里的 ARU 或 CPU 设定的两个控制位，CCUx 触发器信号 TRIG_CCU0 和 TRIG_CCU1 产生边沿。

注释：更多有关 ATOM 和 SOMC 模式，请参考 AURIX™用户手册。

2.3 GTM 路由机制-先进的路由单元(ARU)

GTM 子模块之间的通讯机制是它的一个关键元素,尤其是 GTM 需要独立于 CPU 之外而工作时。

例如一个输入信号，可以在输入子模块 TIM（定时器输入模块）上被捕获到并被特征化。结果被传输到 MCS（多通道序列）以处理数据并触发一个 ATOM 生成输出信号。

这两个通讯路径都由中央 ARU 引导。路由数据的宽度是 53 位；数据大小是 2x 24 位，且其中有 5 个位可被用作 ARU 控制位（ACB）。

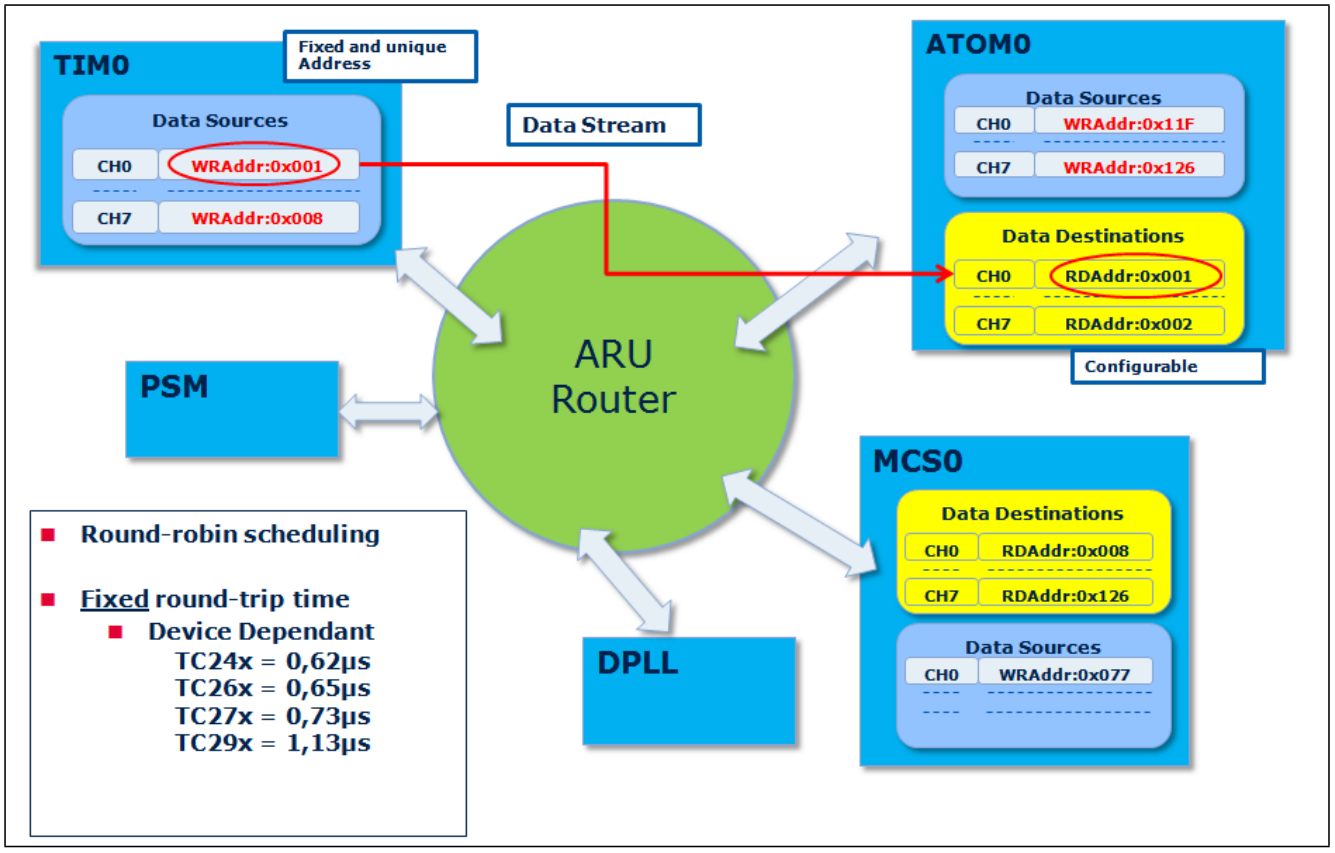

图 5　ARU（先进路由单元）

ARU 阻塞机制是 GTM 一个重要的结构特性。有了这个机制，只要通道未获取新的输入数据，就可能保持在非有效状态。

例如，MCS 保持阻塞，直至 TIM 通道接收到并处理了新的输入数据。因此无需 MCS 中断来通知 MCS 通道有新的输入数据，由于 MCS 只是简单的接受了路由器提供的来自 TIM 通道的新数据，并接着继续工作，。结果是不需要一个内部中断处理机制和控制器。

注释：有关 ARU 的更多信息，请参考 AURIX™ 用户手册。

2.4 可编程多任务硬件核 MCS

多通道序列(MCS)是一个强大的 GTM 子模块。它在位于本地 RAM 里的一个类似于 RISC 的指令集上运行，可以同时是 ARU（先进路由单元）上的信号源和信号目的地。

MCS 的汇编程序必须在系统启动时由 CPU 通过 MCS 总线接口加载到本地 RAM 里去。

CPU 也可以控制 MCS 的内部状态并且访问所谓“任务”的本地寄存器。这些任务的指令存放于 RAM，是可独立执行的线程。因为每个任务都有自己独立的编程计数器和寄存器组，因此任务可同时在同一个程序或不同的程序上运行。

与 GTM 内的所有其它子模块一起，MCS 可被加倍从而获取更多的计算性能。

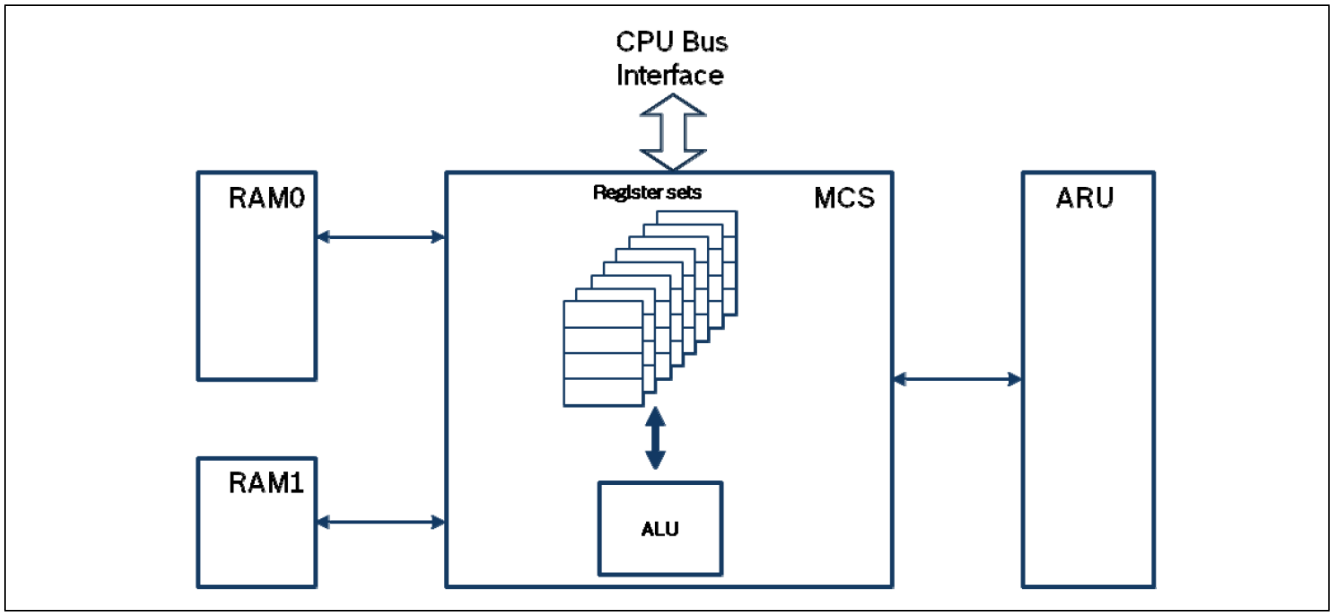

图 6　MCS 架构

MCS 性能包括：

- 在 100MHz 子模块时钟下，单个任务启用高达 20MIPS，(8 个任务启用大约 12.5 MIPS)
- 8 个独立的任务
- 两个不同的任务调度机制
  - -加速的调度机制
  - -循环的调度机制
- 两个独立的 RAM 接口，以便 MCS 任务和 CPU 同时访问两个独立的 RAM 模块。
- 共享指令和数据 RAM
- 高达 8KB 可编址的 RAM 空间
- 每个任务高达 8 个通用寄存器（GPRs）
- 可由 MCS 指令触发的每个任务专用中断

注释：更多有关 MCS 的信息，请参阅 AURIX™ 数据手册。

3 基本脉冲产生

3.1 介绍

连接 ARU 的定时器输出模块(ATOM),是 GTM 负责产生复杂信号的一部分,因此它是用于 MPG 驱动程序的最重要的 GTM 模块。

每个 ATOM 可在不同模式下操作。在本文件中，我们关注信号输出比较（SOMC)模式。

每个 ATOM 模块包含了 8 个通道。每个通道之间都可以相互独立工作，以产生输出信号。

一般来说，每个 ATOM 通道都可被 3 个可用的 CPU 之一控制或者是被多通道定序器（MCS)任务之一控制。

3.1.1 ATOM 受 CPU 控制

当 ATOM 受 CPU 控制时，为了设置脉冲的开始和结束情况，要管理若干中断。CPU 也需要执行不同种类的脉冲所要求的基本计算。

中断通讯量和 CPU 工作负荷通常不是 AURIX™ 产品家族会面临的问题。然而，根据并行管理的脉冲数，以及在微控器上运行的其它应用，基于 MCS 的一种解决方案可以增加系统效率并使 CPU 远离重复性的工作。

3.1.2 ATOM 受 MCS 控制

使用多通道序列（MCS)，就有可能通过 ARU 控制任何可用的 ATOM 通道。这个方法非常有效，因为在理论上无需中断就可管理/设置脉冲开始和结束情况，并且无需 CPU，仅需定序器就能执行所有的基本计算。

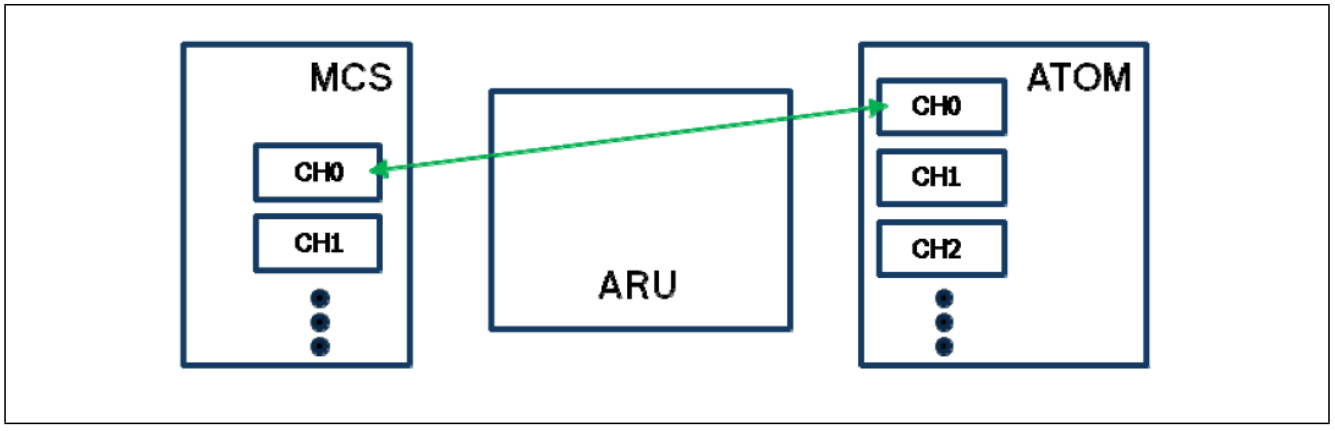

图 7 ATOM 被 MCS/ARU 控制

然而，由于仅能在汇编器里编程 MCS，因此采用这种方法会增加代码的复杂性。

3.2 角度-角度脉冲

可以产生单个脉冲,同时指定该脉冲的绝对开始角度 (AngStart)和绝对终止角度 (Ae)。作为一种约束条件，也可以指定一个截止时间(Tm)。

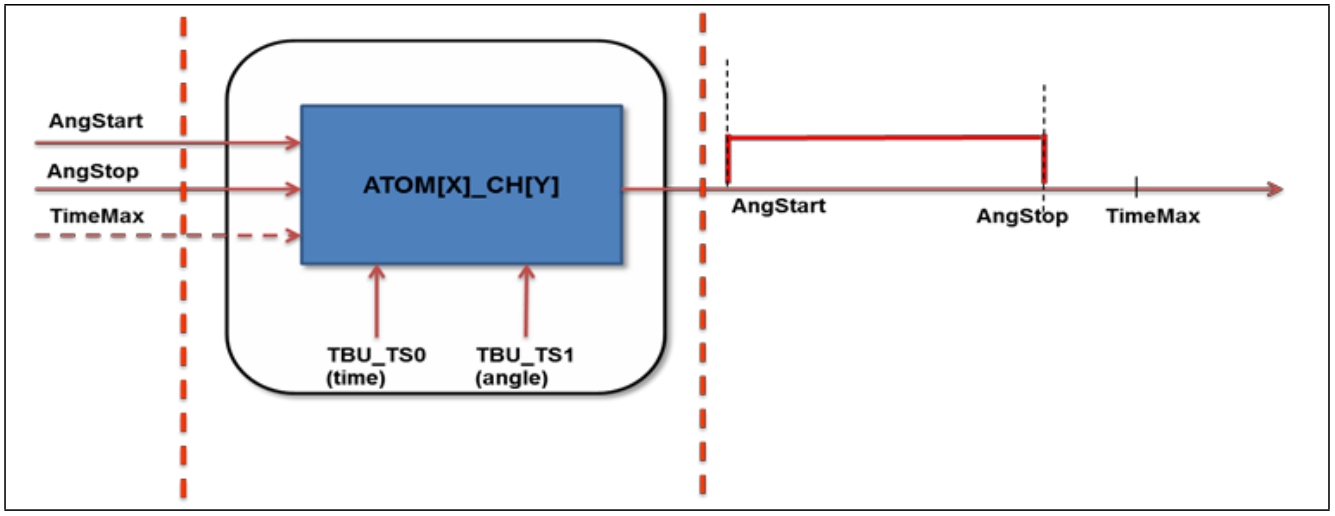

图 8 角度-角度系统概览

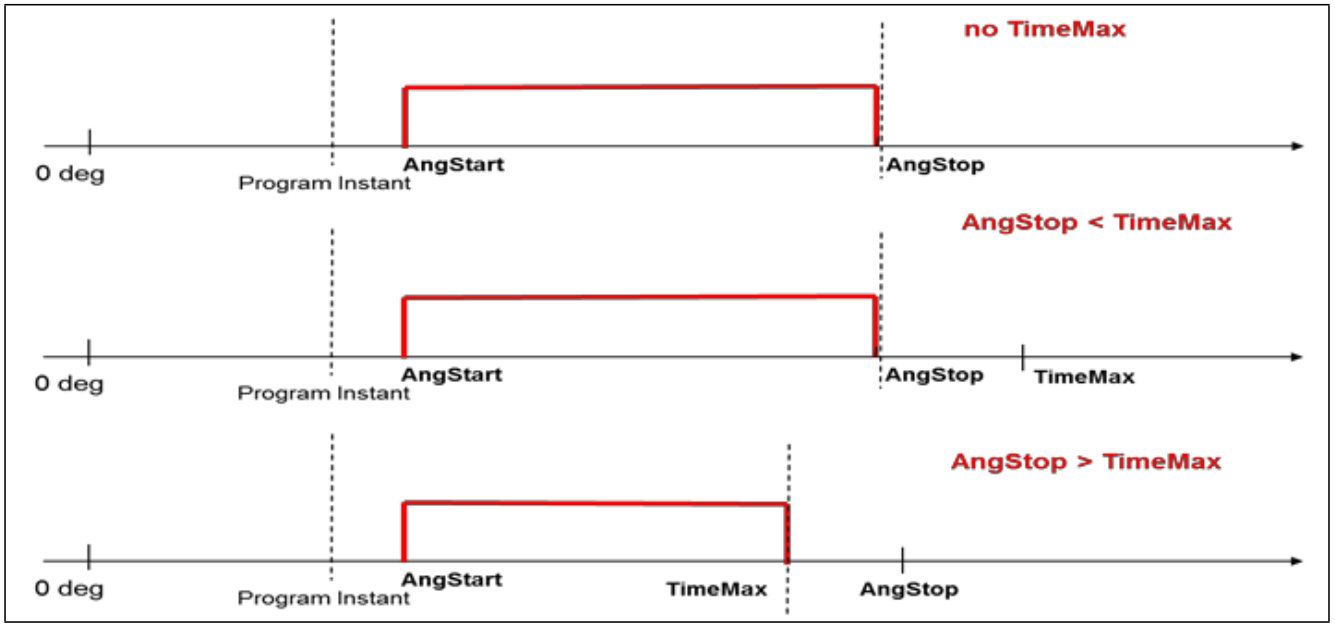

图 9 角度-角度脉冲

开始/结束角度，以及任何所需的时间限制，都要被分别插入到 ATOM CM1(角度）和 CM0(时间）寄存器中。可以使用 CPU 控制 ATOM，或者使用一个特定的 MCS 控制脉冲的产生。

3.2.1 角度-角度受 CPU 控制

第一步是设置开始条件(Start Angle)。

TBU_CH1 计数器（角度计数器）提供角度基准。使用 ACB 位，就有可能控制 ATOM CCU0/CCU1 比较行为和 CCU0/CCU1 匹配上所需的输出。

在这种情况下（角度-角度），GTM_ATOMx_CHy_CTRL.ACB 需被设置为 0xD，这意味着：

•使用 CCU1 仅仅是为了比较（角度）
•在匹配时激活输出

CCU0 / CCU1 输出行为被最后两个 ACB 位，信号本身或者是在 GTM_ATOMx_CHy_CTRL.SL 位里设置的有效电平控制。

SL	ACB10(5)	ACB10(4)	Output behaviour
0	0	0	No signal level change at output (exception in table 12.3.2.2.1 mode ACB42=001)
0	0	1	Set output signal level to 1
0	1	0	Set output signal level to 0
0	1	1	Toggle output signal level (exception in table 12.3.2.2.1 mode ACB42=001)
1	0	0	No signal level change at output (exception in table 12.3.2.2.1 mode ACB42=001)
1	0	1	Set output signal level to 0
1	1	0	Set output signal level to 1
1	1	1	Toggle output signal level (exception in table 12.3.2.2.1 mode ACB42=001)

图 10 ATOM SOMC 输出行为状态

在本例中，ACB=0x0D ($0x01101_b$)，这意味着：

- ACB[8]=0
- ACB[7]=1
- ACB[6]=1
- ACB[5]=0
- ACB[4]=1

下表中的 ACB [5] 和 ACB [4]规定 SL=0,输出应该被设置为 1(第二行)；SL=1，输出则应被设置为 0(第 6 行)。

如图 11 所示,其余的 3 个 ACB 位控制 CCUx 捕获/比较策略。

ACB42(8)	ACB42(7)	ACB42(6)	CCUx control
0	0	0	Serve First: Compare in CCU0 using *TBU_TS0* and in parallel in CCU1 using *TBU_TS1* or *TBU_TS2*. Disable other CCUx on compare match. Output signal level on the compare match of the matching CCUx unit is defined by combination of SL, ACB10(5) and ACB10(4). Details see table 12.3.2.2.1
0	0	1	Serve First: Compare in CCU0 using *TBU_TS0* and in parallel in CCU1 using *TBU_TS1* or *TBU_TS2*. Disable other CCUx on compare match. Output signal level on the compare match of the matching CCUx unit is defined by combination of SL, ACB10(5) and ACB10(4). Details see table 12.3.2.2.1
0	1	0	Compare in CCU0 only, use time base *TBU_TS0*. Output signal level is defined by combination of SL, ACB10(5) and ACB10(4) bits.
0	1	1	Compare in CCU1 only, use time base *TBU_TS1* or *TBU_TS2*. Output signal level is defined by combination of SL, ACB10(5) and ACB10(4) bits.
1	0	0	Serve Last: Compare in CCU0 and then in CCU1 using *TBU_TS0*. Output signal level when CCU0 matches is defined by combination of SL, ACB10(5) and ACB10(4). On the CCU1 match the output level is toggled.
1	0	1	Serve Last: Compare in CCU0 and then in CCU1 using *TBU_TS1* or *TBU_TS2*. Output signal level when CCU0 matches is defined by combination of SL, ACB10(5) and ACB10(4). On the CCU1 match the output level is toggled.
1	1	0	Serve Last: Compare in CCU0 using *TBU_TS0* and then in CCU1 using *TBU_TS1* or *TBU_TS2*. Output signal level when CCU1 matches is defined by combination of SL, ACB10(5) and ACB10(4).
1	1	1	Not used when ARU disabled.

图 11 ATOM SOMC 捕获/比较行为状态

在这个案例中，ACB[8]=0, ACB[7]=1, ACB[6]=1，且仅仅使用了 CCU1(第 3 行）。

需要将开始角度的情况(startAng)复制到 GTM_ATOMx_CHy_CM1 寄存器里。

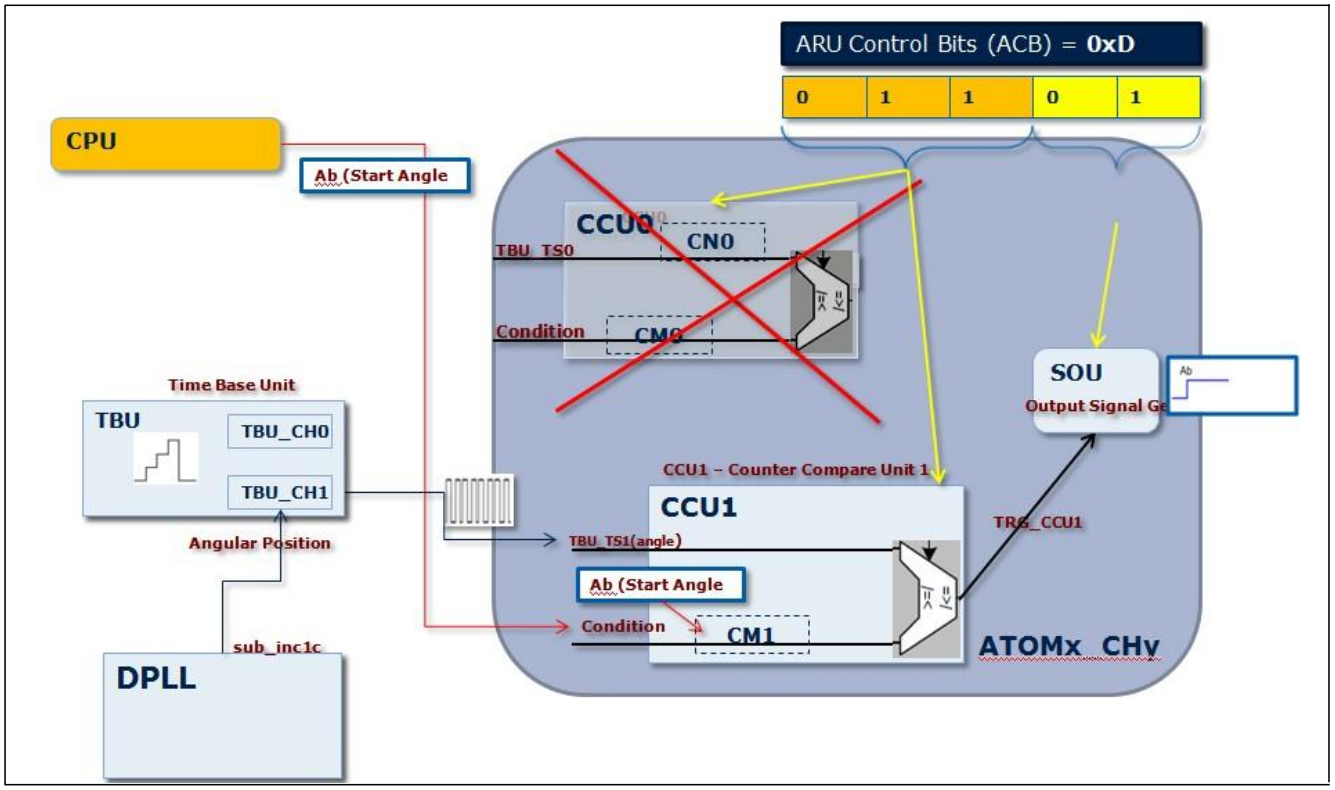

图 12　角度-角度脉冲：startAngle 情况的设置

一旦 TBU_CH1 计数器与储存在 CM1 寄存器里的开始角度匹配上(TBU_CH1 >= CM1):

•TRG_CCU1 中断触发[1]
•输出被设置为 HIGH / ACTIVE 状态
•系统时间值被储存在 ATOM SR0 里
•角度被储存在 ATOM SR1 寄存器里

如果需要在定义的终止角度(<stopAng>)上关闭脉冲，那么需按照相同的方式配置 ATOM(在 CCU1 中断上），但需在<stopAng>上关闭输出。

ACB 值应为 0xE(仅仅 CCU1,在匹配时关闭输出),且需要在 CM1 寄存器里加载<stopAng>参数。

请记住在匹配之后，ATOMx_CHy 被停用。如果要再次将其使能，就需要读/访问包含了捕获的 TBU_CH1 值的影子寄存器 SR1(SR0 寄存器包含了系统时间值 TBU_CH0)。

[1] 为了设置停止状态（stopAngle 或者是 stopAngle 和 Tm 间的最小值），有必要使能 CCU1 中断。

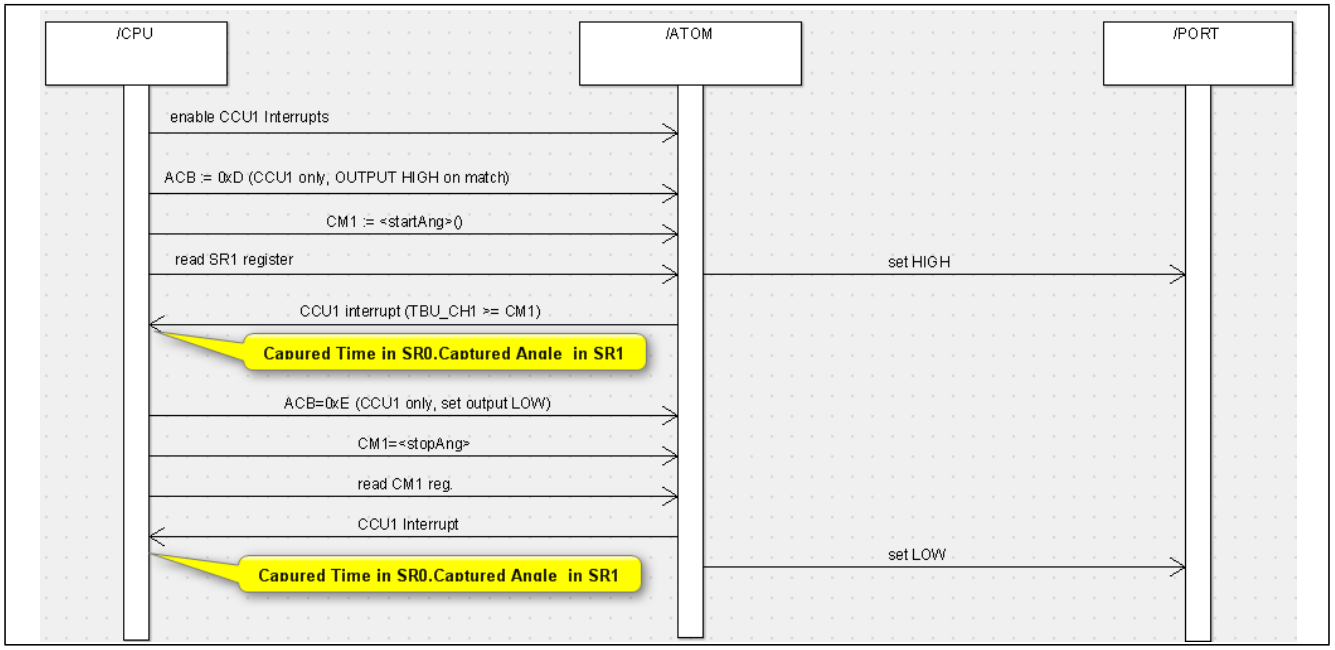

图 13　角度-角度脉冲产生时序图（无截止时间）

当脉冲结束条件也需要截止时间（Tm）时，ATOM 需要被设置为“优先服务”模式(ACB=0x2)，其中两个 CCU 并行工作：

•CCU0 比较截止时间（Tm 保存在 CM0 寄存器）
•CCU1 比较截止角度（<stopAng>保存在 CM1 寄存器）

当 CCUx 在比较匹配时被停用。

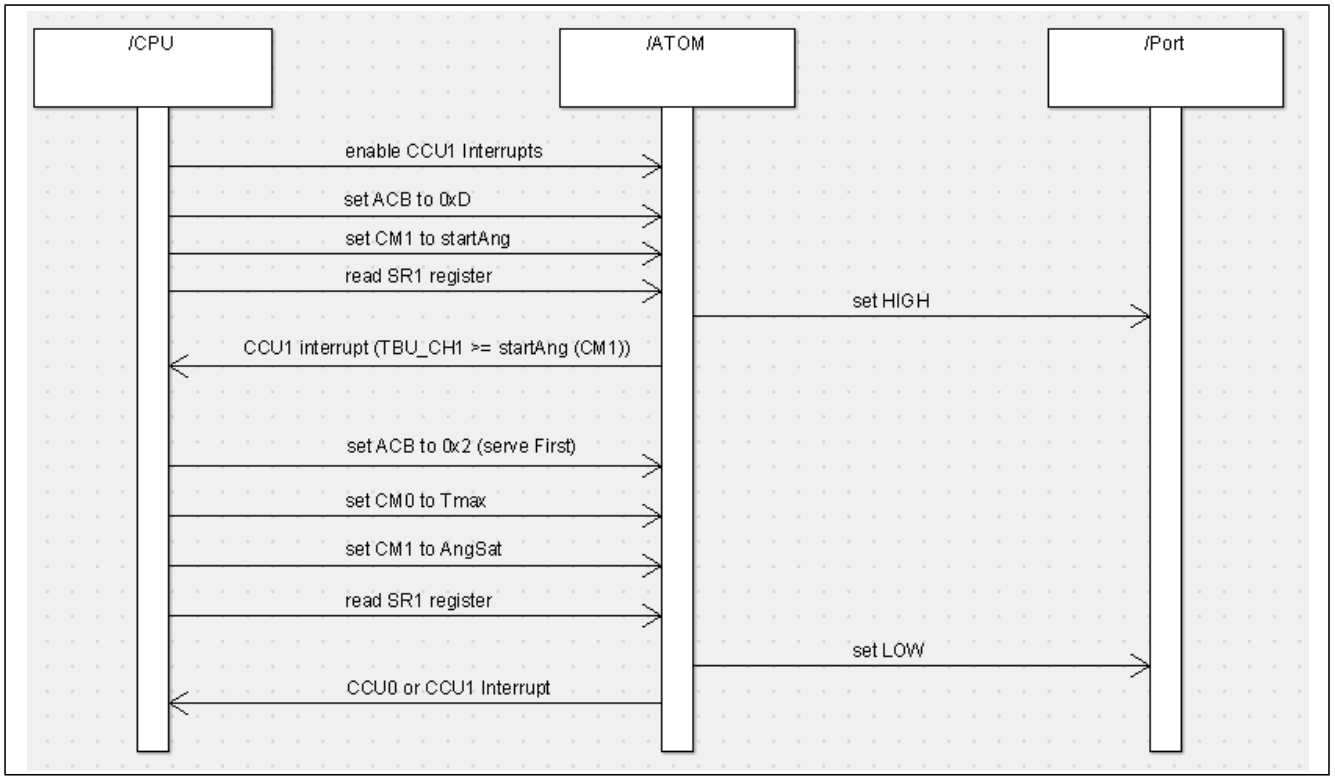

图 14　角度-角度脉冲产生时序图（有截止时间）

3.2.2 MCS 控制角度-角度

通过使用其中一个可用的指令序列通道(MCS),就可能卸载 CPU。这样就不再需要中断第一次匹配(<startAng>)。下列伪代码是 MCS 汇编代码的一个示例，控制 ATOM 产生角度-角度脉冲。

```
csg_AA_Ab_Ae_Tm:
          ;get start parameter, R3 <= Ab (Start Angle)
        movl R0 START_ANGLE
        mrdi R3 R0
        ;load ARU Write address in R6<= for indirect ARU write
        movl R0 ARU_WR_ADDR
        mrdi R6 R0
        ;program ATOM channel with 1st event.(Angle Begin)
        movl acb $D ;Compare in CCU1 only, activate output on capture
        awri R3 R3 ; ARU Blocking Write
          ;load ARU Read address in R6<=for indirect ARU write
        movl R0 ARU_RD_ADDR
        mrdi R6 R0
        ardi R2 TBU_TS1 ; Wait for capture (Match on Ab)
        ;get end parameter (Ae), R3<= Stop Angle
        movl R0 STOP_ANGLE
        mrdi R3 R0
        ;get max time parameter (Tm), R4<=Tm
        movl R0 TM_MAXTIME
        mrdi R4 R0
        add R4 R2 ;captured start time + maximum duration
        ;load ARU Write address in R6<= for indirect ARU write
        movl R0 ARU_WR_ADDR
        mrdi R6 R0
         ;program ATOM channel with 2nd event.
        movl acb $2 ;serve first (Ae/Tm) => deactivate output on compare
        awri R4 R3
         ;load ARU Read address in R6<=for indirect ARU write
         movl R0 ARU_RD_ADDR
        mrdi R6 R0
        ardi R3 R1 ; Wait for capture
......
csg_AA_Ab_Ae_Tm_end:
```

图 15 代码片段 角度-角度汇编代码

这个示例考虑了时间 (Tm)限制。如果真实情况与该示例有差异，为了仅考虑<stopAng>，则应该修改结束条件。

3.3 角度-时间脉冲

角度-时间脉冲是从特定的开始角度(<AngStart>)开始并具有特定长度的脉冲(<PulseLen>)。
需要计算结束时间<Tstop>以确定脉冲关闭的瞬间。

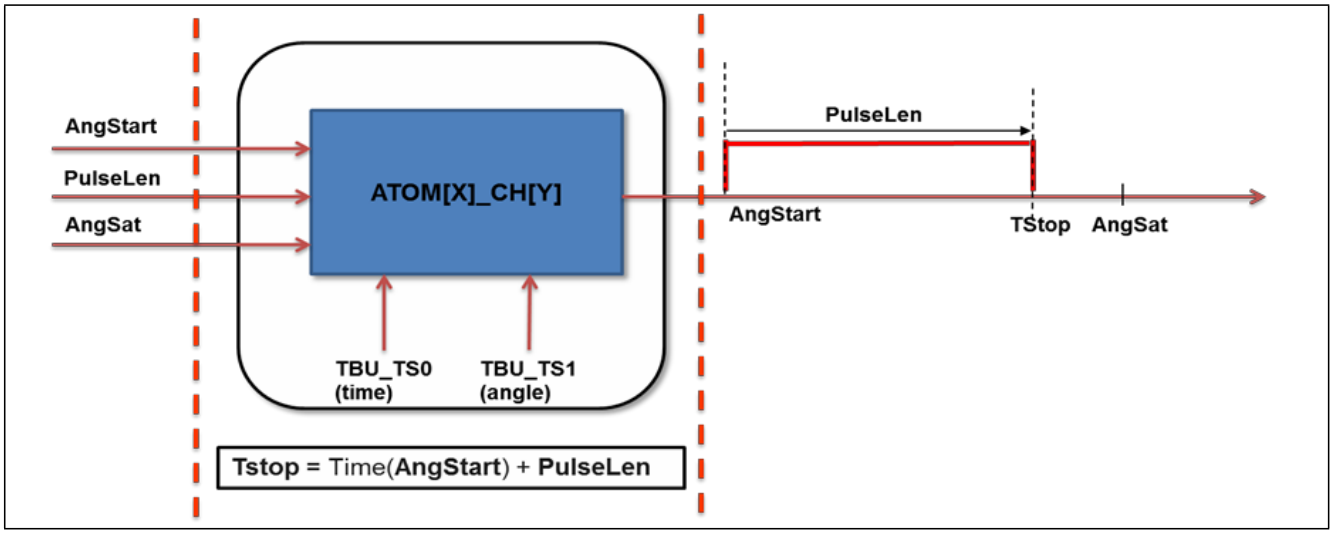

图 16 角度-时间脉冲系统概览

可以指定一个脉冲结束饱和角度<AngSat>，以在<TStop> > <AngSat>时，迫使脉冲在规定的脉冲长度之前关闭。
因为脉冲是以角度开始的，因此不需要对开始角度重新进行编程。

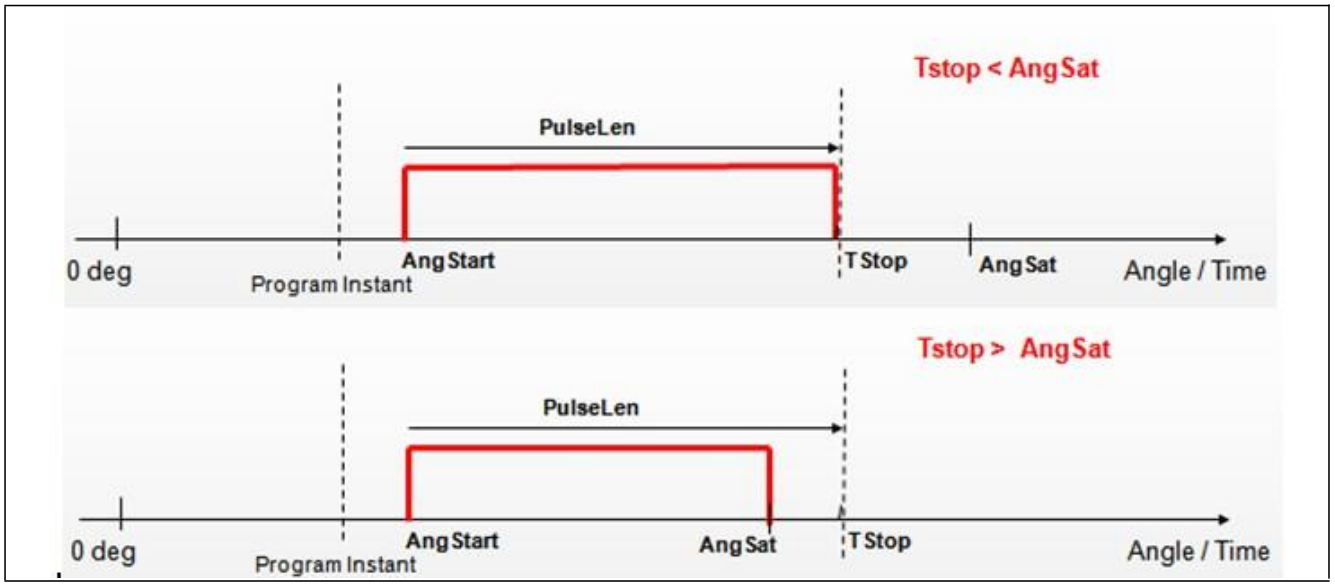

图 17 角度-时间脉冲

3.3.1 角度-时间产生受 CPU 控制

在角度-时间脉冲里,<AngStart> 条件的设置和我们在考虑角度-角度脉冲时一样。唯一的一点差别是，在与<AngStart>匹配时，CPU (MCS)需要使用如下公式计算<TStop>：

- Tstop = T(AngStart) +<PulseLen>

T(AngStart)是 TBU_CH1 与<AngStart>相匹配时的系统时间值。可从寄存器 SR0 中获得该值。

如同在“角度-角度”章节的描述，在计算<Tstop>之后，ATOM 需被设置为“优先服务”模式。

下列时序图是所需操作的概览图：

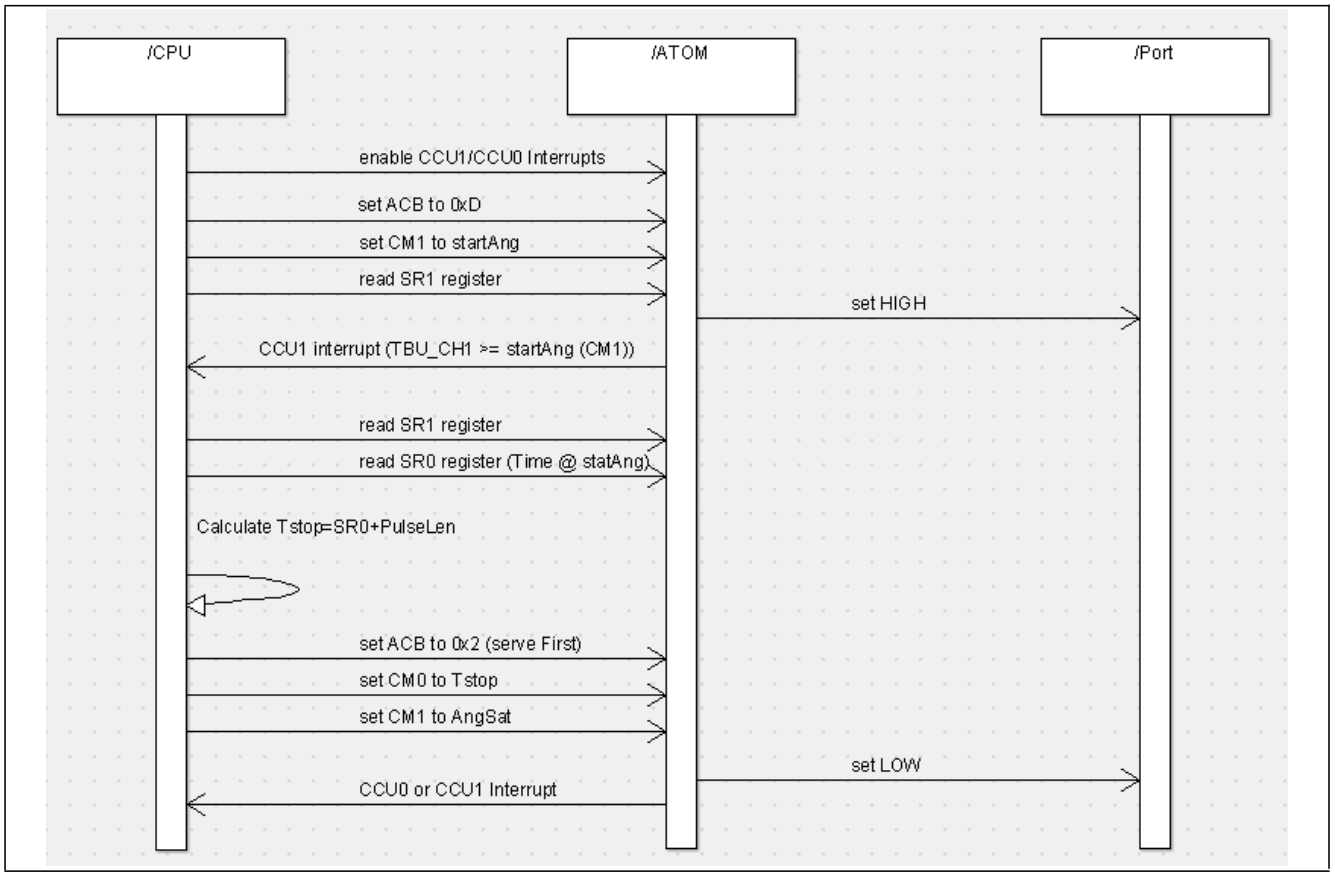

图 18 角度-时间脉冲产生的时序图

3.3.2 角度-时间产生受 MCS-ARU 控制

通过使用其中一个可用的指令序列通道，就可以将 CPU 从一些计算中解放出来，并且在发生首次匹配(<startAng>)时不再需要中断。

下列伪代码是 MCS 汇编代码的一个示例，控制 ATOM 产生角度-时间脉冲。

```
;AT_Ab_Pw_Am pulse generation function
;Sequence:
;         - Program Ab (AngStart)
;         - wait for capture
;         - Compute Te (TStop)
;         - program Te/Am (Am=AngSat) with acb = 0x3 serve -First)
;         - wait for capture
csg_AT_Ab_Pw_Am:
          ;R3←Ab (AngStart)
          movl R0 START_ANG
          mrdi R3 R0
        ;R6 <= ARU Write address in R6 for indirect ARU write
        movl R0 ARU_WR_ADDR
        mrdi R6 R0
        ;program ATOM channel with 1st event (AngStart).
        movl acb $D ;Compare in CCU1 only, activate output on capture
        awri R3 R3     ;Write AngStart to ATOM (in R6 Atom Address)
          ; R6 <= load ARU Read address in R6 for indirect ARU write
        movl R0 ARU_RD_ADDR
        mrdi R6 R0
        ardi R2 TBU_TS1 ; Wait for capture (ARU blocking Read)
        ;R3←PulseLen

        movl R0 PULSE_LEN_ADDR
        mrdi R3 R0
            ;Compute (TStop)
        add R3 R2 ;R2 contains comp cond.on CCU0(capt start
                  ; time,T(AngStart))
        ;R6 <= ARU Write address in R6 for indirect ARU write
        movl R0 ARU_WR_ADDR
        mrdi R6 R0
          ;program ATOM channel with 2nd event.
        movl acb $2 ;serve first (Te/AM) => deactivate output on compare
        awri R3 R4
          ; R6 <= load ARU Read address in R6 for indirect ARU write movl R0
        ARU_RD_ADDR
        mrdi R6 R0
```

```
        ardi R3 R1 ; Wait for capture
        ;compute active time and add it to the act. time of the curr sequence.
          ;R3 contains TStop.
.........
csg_AT_Ab_Pw_Am_end:
        ret ;end of AT_Ab_Pw_Am pulse generation function
```

图 19　代码片段 2　　角度-时间汇编码

3.4 时间-角度脉冲

在 TimeAngle 模式下,可以产生具有指定的绝对结束角度(<AngStop>)和指定的脉冲长度(<PulseLen>)的单脉冲，该单脉冲可确定脉冲的开始瞬间。

注释：必须保留脉冲长度，因此需要重新编程开始角度。

可以指定一个终止饱和角度 (<AngSat>)，且终止角度永远不可以超过它。

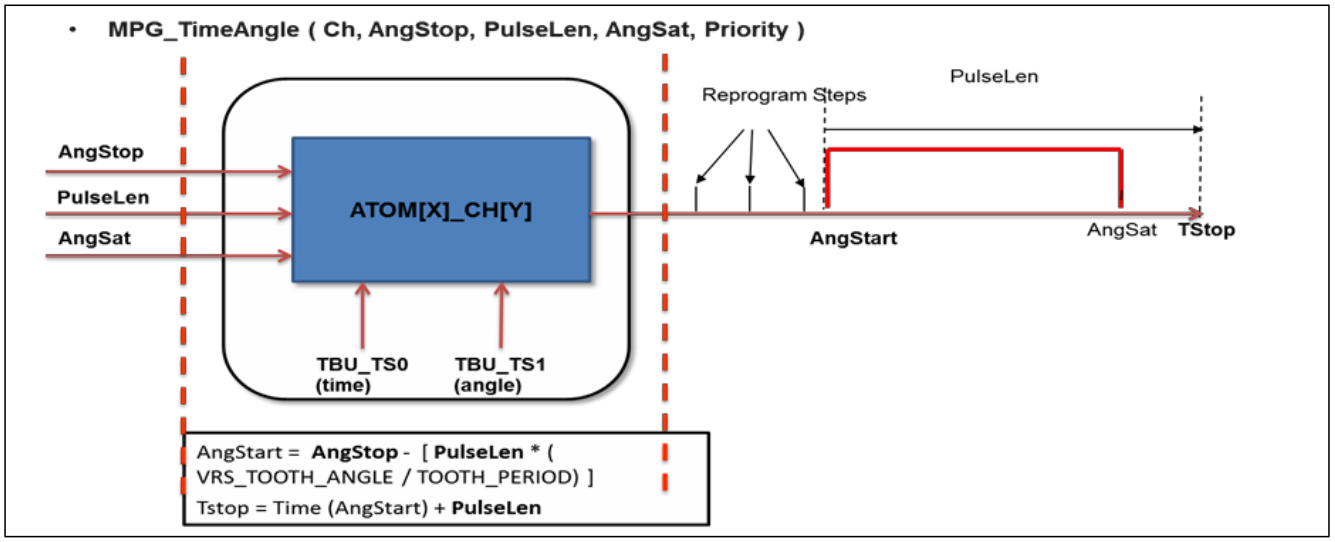

图 20　时间-角度系统概览

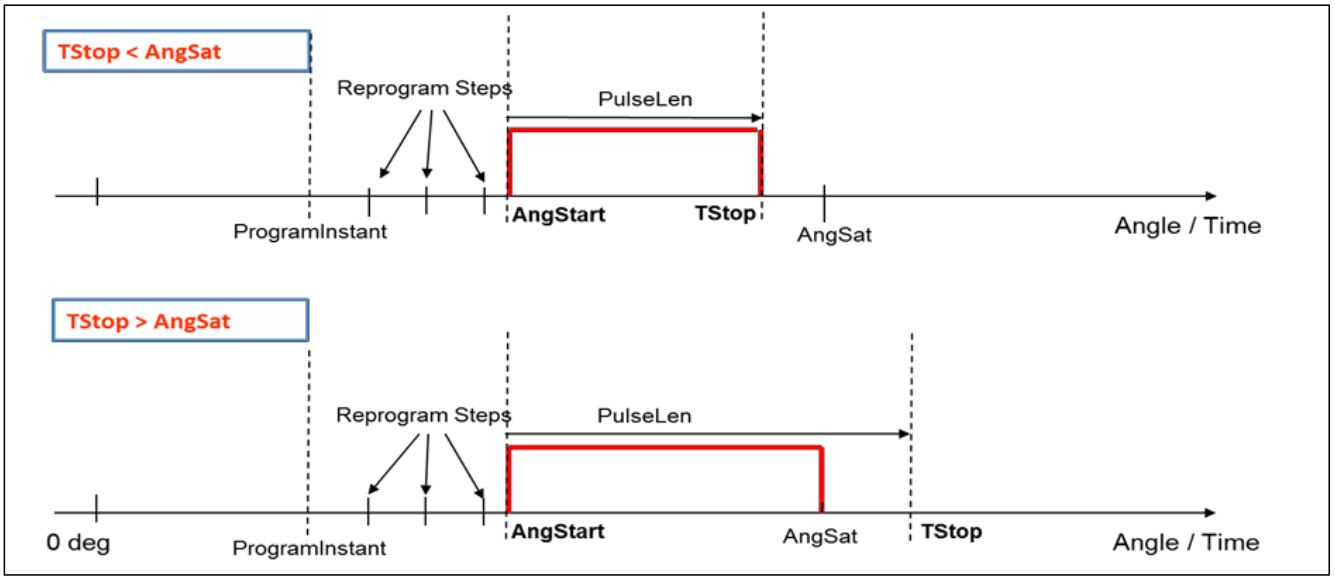

图 21　时间-角度脉冲

3.4.1　开始条件

有不同的方法计算<AngStart>:

1. 中断时 CPU 计算
2. DPLL PMTR(位置减时间请求，需 ARU)

3.4.2 使用 CPU 计算 “Start Angle”

下列程序图显示了使用 CPU 计算开始角度的所有操作：

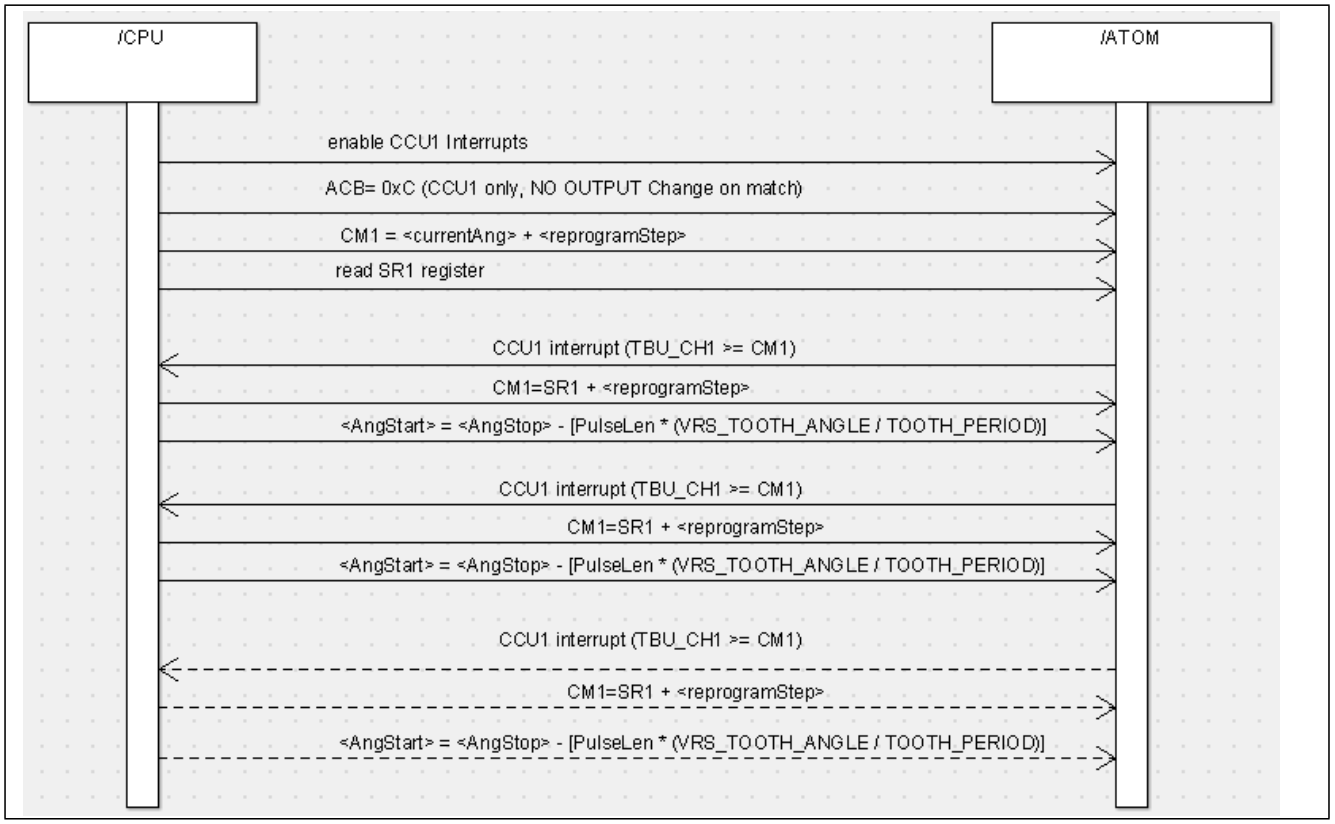

图 22 使用 CPU 计算 AngStart

驱动程序应周期性的计算开始和结束角度，直至当前的角度(TBU_CH1)接近开始角度（脉冲开始前，MPG_REPROG_START_ANGLE 角步长）

每次重设都应出现在 MPG_REPROG_STEP_ANGLE 角步长定义的空间内，直至当前的角度接近于开始角度 MPG_REPROG_STOP_ANGLE（最后一个重新编程点）。

在进行最后一步设程序后，无法再对开始和结束角度进行修改，并且脉冲长度随着脉冲执行过程中的加速度/减速度变化。

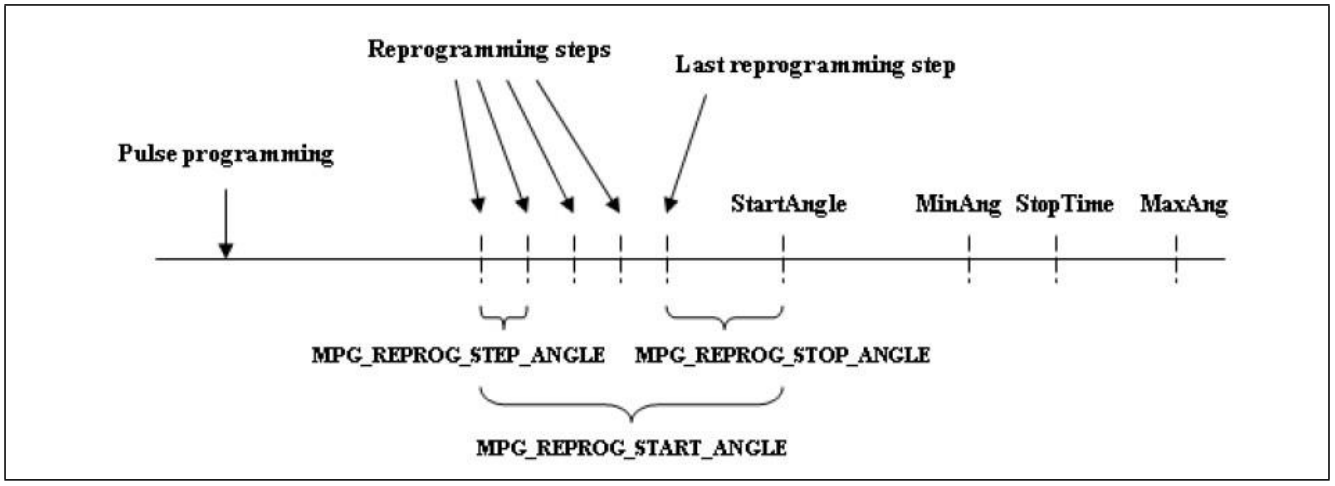

图 23 时间-角度脉冲重编程序逻辑

3.4.2.1 使用 DPLL 计算“Start Angle”

如果已知<PulseLen>和<AngStop> (相对),GTM DPLL 可以估算预期的开始角度，同时考虑加速度和减速度。该操作被称之为位置减去时间（PMT）。

为了使用 DPLL 的这个特性，需要通过 ARU 将 ATOM 连接到 DPLL。在每一个齿上，DPLL 更新开始角度并通过 ARU 自动给 ATOM CM1 寄存器编程。

为了实现这个方案，一般可以使用特定的汇编码给 MCS 编程：

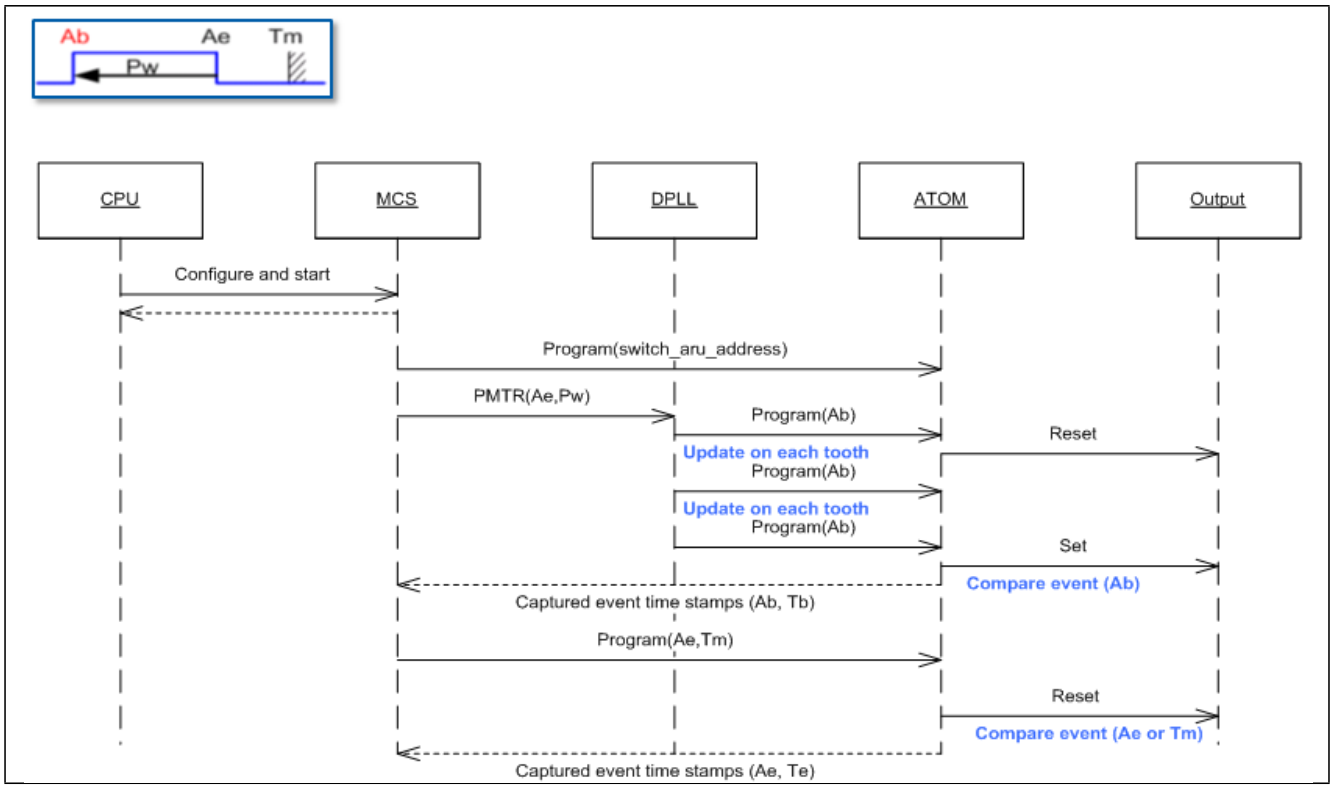

图 24　使用 DPLL/MCS 计算 AngStart

```
;AT_Ae_Pw_Am pulse generation function for a pulse that should stop at a time Te (TStop)
; calculated from the angle begin Ab and the pulse width Pw.Ab must be calculated from
Ae, Pw and
; the current speed.Ab is calculated by the the DPLL and is updated every tooth.
;Sequence:
;       - load Ae and Global Angle Offset (relative 0°  (TDC of cylinder_1)
of the current window)
;         - Load PW
;         - Inform ATOM to expect next compare value from the DPLL
;         - Send a PMTR to the DPLL (StopAng(Ae), PulseLen(Pw))
;         - wait for capture
;         - program Te/Am (Close Condition)
csg_AT_Ae_Pw_Am:
          ;the acb is forwarded to the ATOM by DPLL
```

```
          movl R5 $D ;Compare in CCU1 only, activate output on capture
          jmp csg_AT_Ae_Pw_Am_start
          csg_AT_Ae_Pw_Am_start:
         ;R3← StopAngle (Ae)
          movl R0 STOP_ANGLE
        mrdi R3 R0;
        call  csg_atom_aru_switch      ;ATOM shall  receive  the  next  data  from
DPLL

        ;R4<= PulseLen (Pw) - load pulse width
        movl R0 PULSE_LEN
        mrdi R4 R0
       ;load DPLL ARU write address in R6 for indirect ARU write
        movl R0 DPLL_WR_ADDR
        mrdi  R6  R0
        mov  acb  R5
        awri R4 R3
        ;load ARU Read address in R6 for indirect ARU write
        movl R0 ATOM_ARU_RD_ADDR
        mrdi R6 R0
        ardi R2 TBU_TS1 ; Wait for capture
        ;Second Part ➔        Close Condition
        ;get max angle parameter SatAng (Am)
        movl R0 SAT_ANGLE
        mrdi R3 R0
        ;compute TStop (Te)
          add R4 R2 ;TStop=captured start time + PulseLen
        ;load ARU Write address in R6 for indirect ARU write
        movl R0 ARU_ATOM_WR_ADDR
        mrdi R6 R0
          ;program ATOM channel with 2nd event.
        movl acb $2 ;serve first (TStop/SatAng) => deactivate output on
compare
        awri R4 R3
          ;load ARU Read address in R6 for indirect ARU write
        movl R0 ATOM_ARU_RD_ADDR
        mrdi R6 R0
        ardi R3 R1 ; Wait for capture
…
……
csg_AT_Ae_Pw_Am_end:
……
……
csg_atom_aru_switch:
     ;load DPLL ARU write address in R6 for indirect ARU write
```

```
      movl R0 DPLL_ARU_WR_ADDR
      mrdi R6 R0
      ;switch atom read address
      movl acb $1C ;force ATOM to read next compare value from DPLL
      movl R0 $0
ret   awri R0 TBU_TS1 ;==>Write to ARU ADDR R6
```

图 25　使用 DPLL/MCS 产生<AngStart>

仅仅使用 CPU 和中断可以实现相同的应用。

3.4.3　结束条件

在<AngStart>匹配后为了创建脉冲结束条件，应将 ATOM 编程为“优先服务”模式，且在首次匹配后，应将输出设置为低。

3.4.4　可变的 PulseLen

在这个应用中，驱动程序需要一个时间-角度脉冲，且相比<AngStop>，<PulseLen>优先级较低。下图显示了这一点。

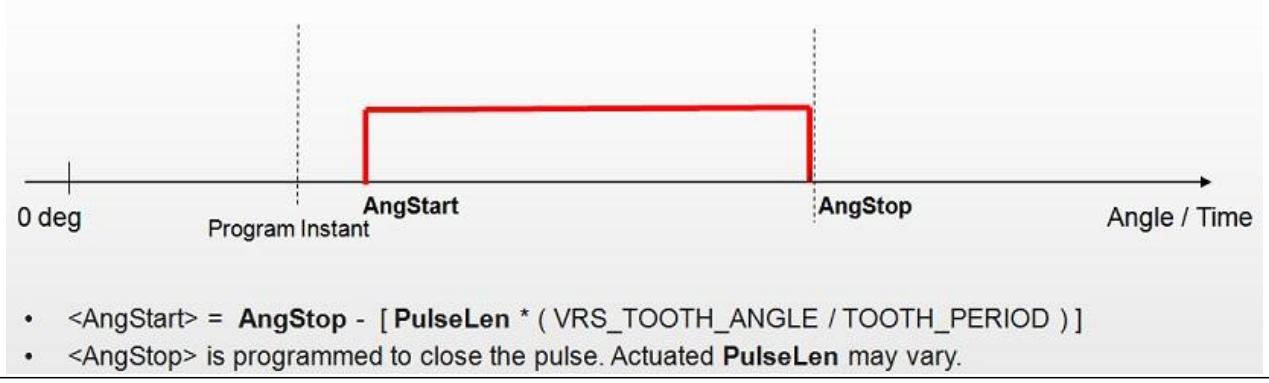

图 26　时间-角度脉冲，PulseLen 优先级低于 AngStop

为了仅使用在 AngStop 上编程的 CCU1，需要改变脉冲结束条件（这一点在角度-角度脉冲示例中也进行了阐述）。

3.5 时间-时间脉冲

3.5.1 介绍

该文件中描述了 2 种不同的时间-时间脉冲。这些脉冲之间的差别在于他们的长度的优先级。尤其是：

•优先级低的脉冲长度（可变的）
  -相比于结束条件，脉冲长度优先级较低，因此在发生加速/减速时，脉冲长度可变化。
•优先级高的脉冲长度
  -应保留脉冲长度（需要重新设定）

注释：　对于优先级高的脉冲长度，在时间-角度章节所考虑到的 DPLL(PMTR)和 MCS 的使用仍然有效。

3.5.2 优先级低的脉冲长度

脉冲在某个规定的时间开始并结束。

注意：PulseLen 随着加速度/减速度而变化。

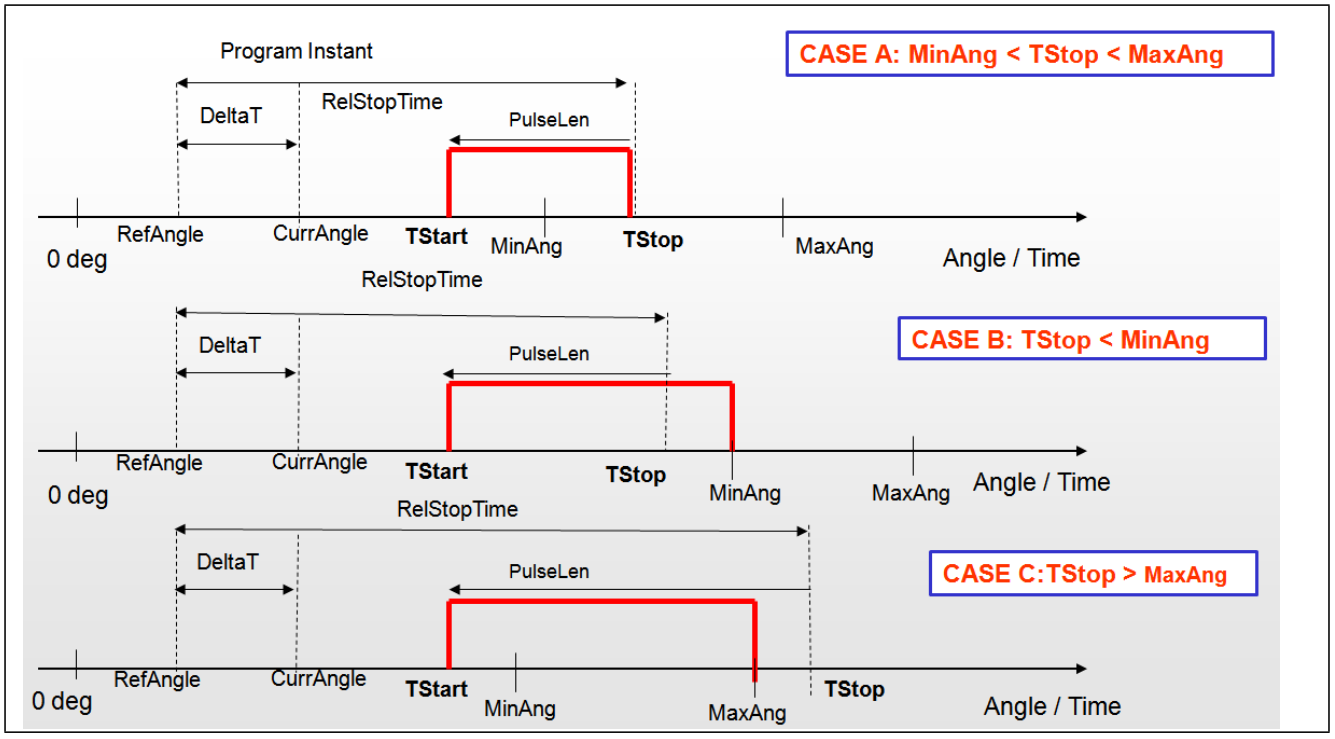

图 27　时间-时间脉冲，变化的 PulseLen

•脉冲应在时间<TStart>开始，并在时间<TStop>结束（例 A）。
•如果在时间<TStop>时，角度小于某个角度(<MinAngle>)，那么应将脉冲延长至<MinAng>（例 B）。
•如果角度在<Tstop>之前达到<MaxAng>，那么应停止脉冲（例 C）。

仅仅在编程瞬间计算<TStart>，这意味着<PulseLen>随着加速和减速而变化。

为了启动补偿以便脉冲正确地参照基准角度
(<RefAngle>),必须执行下列计算:

- DeltaT = (CurrAngle - RefAngle)* (ToothPeriod/VRS_TOOTH_ANGLE)
- TStop = (CurrTime - DeltaT)+RelStopTime
- TStart = TStop-PulseLen

其中:

- DeltaT = 时间补偿
- CurrAng = 编程瞬间的当前角度
- ToothPeriod = VRS 信号一个齿的周期(GTM_DPLL_DT_T_ACT)
- VRS_TOOTH_ANGLE = 每个曲轴齿的微时钟数

3.5.2.1 系统描述

在编程时间时计算<TStart>。

为了建立开始条件(<TStart>)，ATOM 应被设定为仅 CCU0 工作（使用 TBU_CH0 的时间比较）。

下列时序图显示了创建这样一个脉冲所需的所有操作。

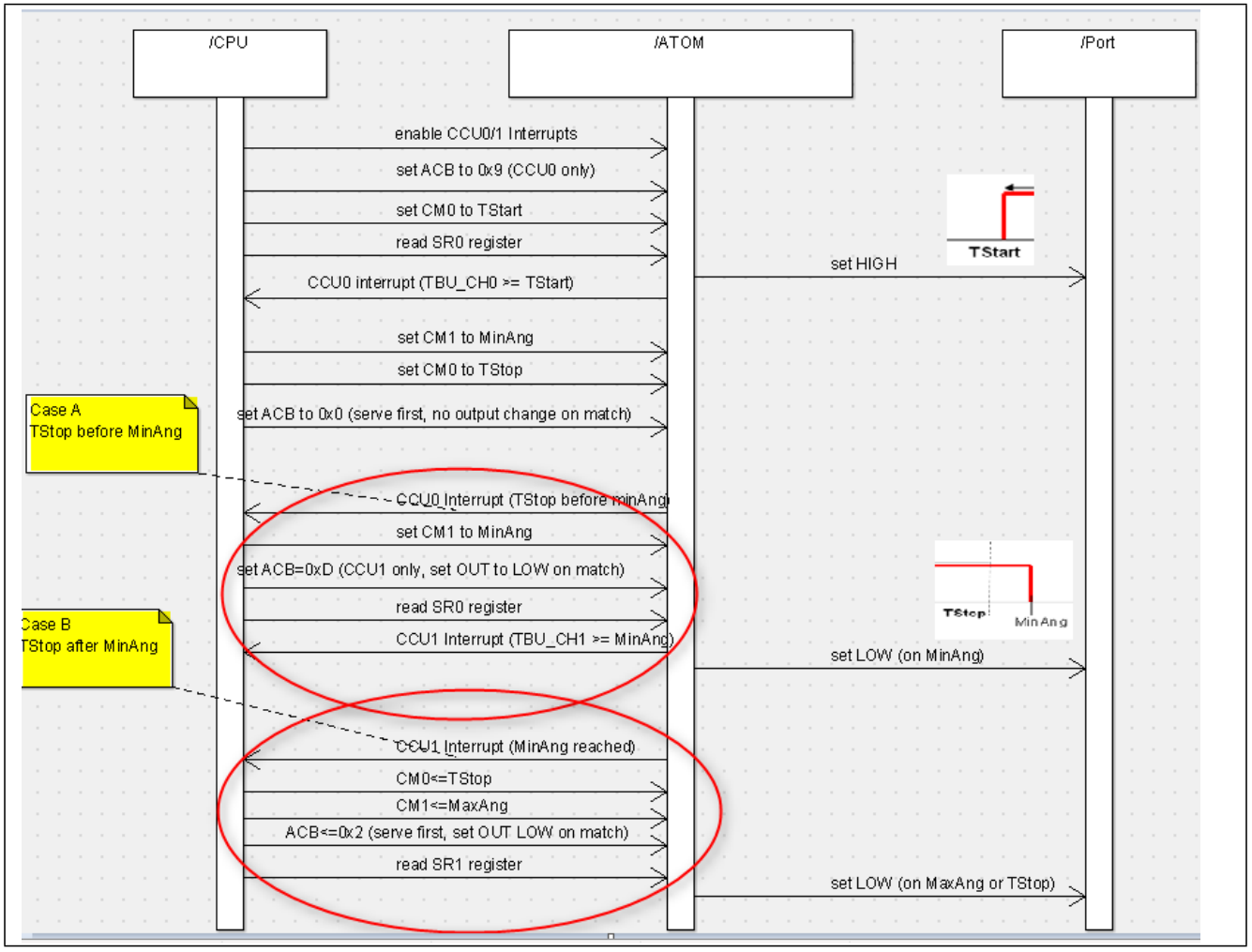

图 28　时间-时间脉冲，变化的 PulseLen

3.5.3 优先级高的 PulseLen

脉冲产生的形式必须使用一些机制来确保脉冲长度优先。 这要求定期重新计算终止角(重新设定)，并在 [<MinAng>-<MaxAng>] 间隔里检查终止角。根据终止角的变化也要重新计算开始角，并保持脉冲宽度。

注释：脉冲以角度开始，并在时间上结束。

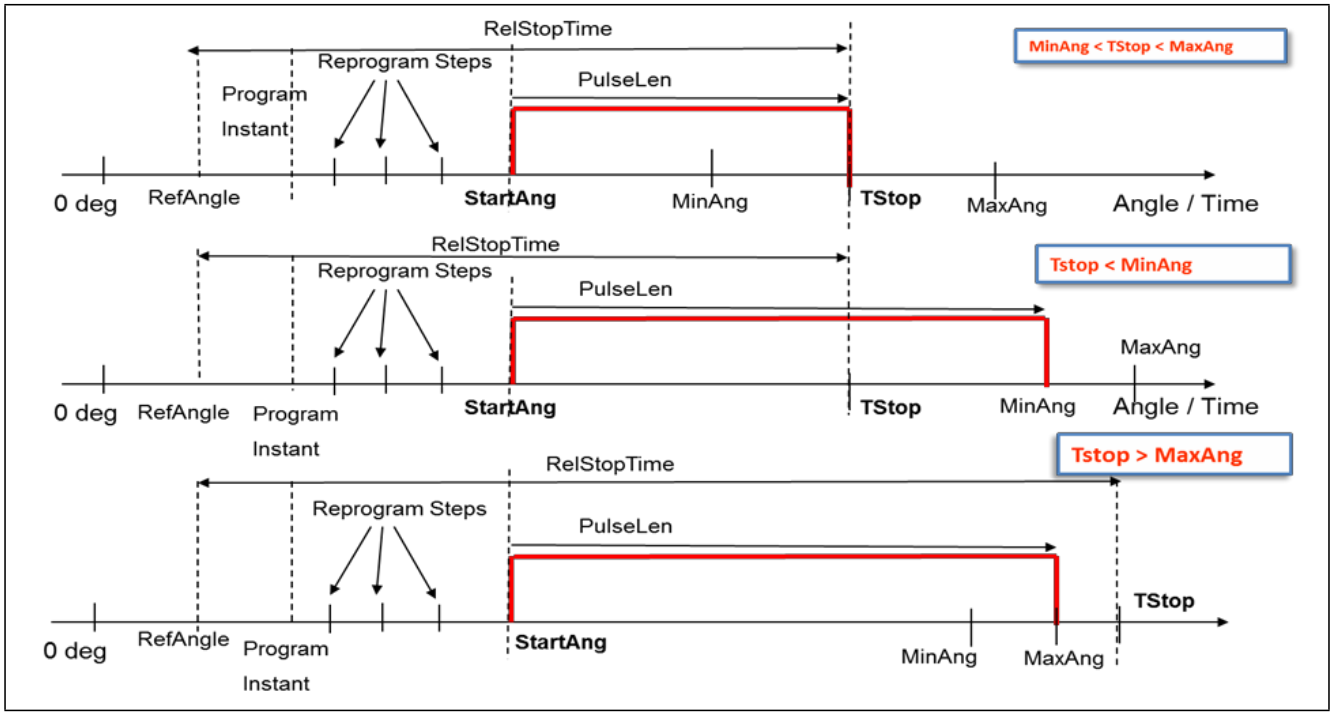

图 29 时间-时间脉冲，不变的 PulseLen

- StopAngle = RefAngle + [RelStopTime * (VRS_TOOTH_ANGLE / TOOTH_PERIOD)]
- StartAng = StopAngle - [PulseLen * (VRS_TOOTH_ANGLE / TOOTH_PERIOD)]
- TStop = Time(StartAngle) + PulseLen

终止角度只是为了校验脉冲关闭限制而计算,但是脉冲在时间上结束。

在脉冲的开始角度，驱动程序必须对终止沿进行编程，以使它在<PulseLen>里建立的一段时间之后出现。因此在该模式下，所有的努力都直接用来获取所需的<PulseLen>。

重新设定策略与第 3.4.2 节使用 CPU 计算 “Start Angle”相同。

3.5.3.1 系统描述

以下流程图显示了使用 CPU 实现时间-时间脉冲所需的所有操作。

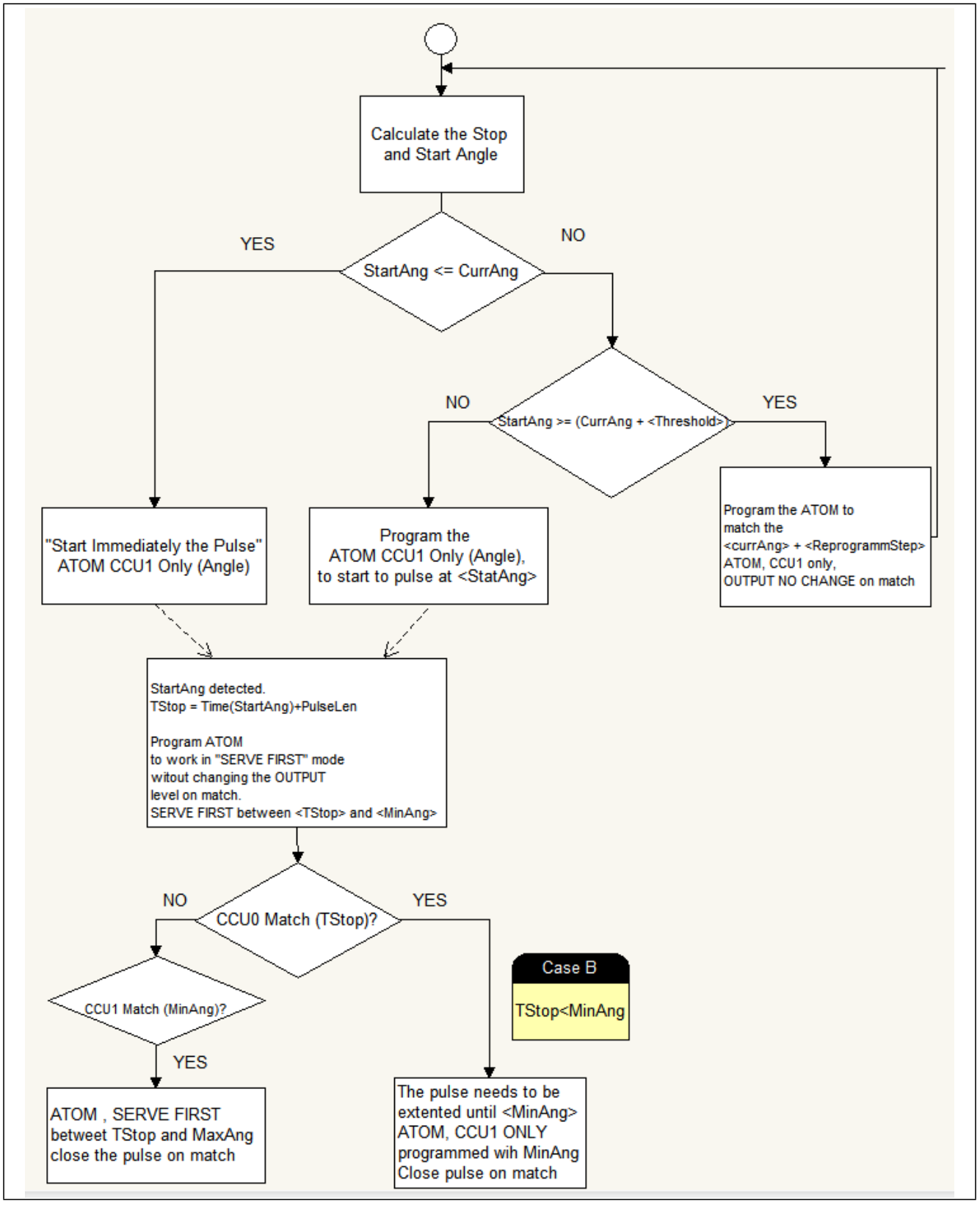

图 30 时间-时间脉冲，优先级高的 PulseLen

3.6 特别注意事项

3.6 相对角 vs.绝对角

为了避免遗漏任何微时钟,并使用 PMTR(位置减时间请求),DPLL(数字锁相环)要求使用了相对角。

角度计数器 TBU_CH1 是一个 24 位的计数器。

一般来说，MPG 驱动程序接口以绝对角工作，角度分辨率为 0.1 度。

例如，为了从 10 度角到 360.5 度角时产生角-角脉冲，可以使用如下参数：

- <AngleStartAbs> = 100 (10.0 degrees)
- <AngleStopAbs> = 3605 (360.5 degrees)

在应用笔记 AP32212 - “GTM 发动机位置驱动程序”里描述的 EP 驱动程序，提供了一系列可从绝对角计算出相对角的泛函数。尤其是，在每次转动之后，EP 驱动程序储存“相对角开使偏移”，这个偏移值加上绝对角度，便得出相对角度。

- <AngleStartRel> = EP_getOffset() + <AngleStartAbs>

因为有 24 位的角度计数器,因此偏移是必须的;在发动机转动时,最大的角度值不是生成的微时钟数的倍数。

比如：

•对于每个齿（6 度），256 个微时钟
•120 个齿=> 256*120 = 30720 个微时钟/720 度
•(0x00FFFFFF+1) / 30720 => 546,13

这意味着由于在每次角度计数器重启之后（转出），相对角度 0x00000000 对应一个不同的绝对角度，因此无法简单地使用一个“模块”操作从相对角度计算出绝对角度（如 relativeAngle>%30720)。

3.6.2 过去/未来效应

MPG 驱动程序的接口以绝对角工作。这意味着，在重新设定 MPG 脉冲时，底层软件总会有一个 720 度长的“angular visibility window”。

现在，我们来想想下列两种情况：

•在编程时间 321.0 角度时的当前角度

•开始角度编写为 320.0 度

在这种情况下，如果应用需要产生喷油脉冲，而这个脉冲的开始角度又已经过去，会发生什么呢？

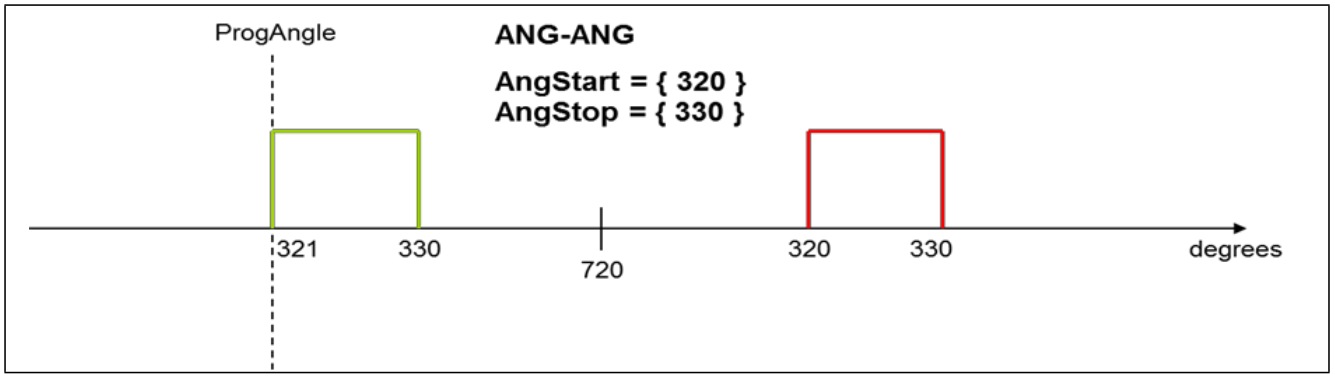

图 31 角度-角度"过去/未来"效应

如果 MPG 驱动程序决定总是在下一转时产生脉冲,那么即使当前角度 (ProgAngle) 比所需的 AngStart 角度大 0.1 度,也可能遗漏喷油。

所采用的逻辑取决于具体的应用。

此处这个 MPG 的实现按照下图中给出的规则，决定脉冲应该在当前这转还是下一转时开启。

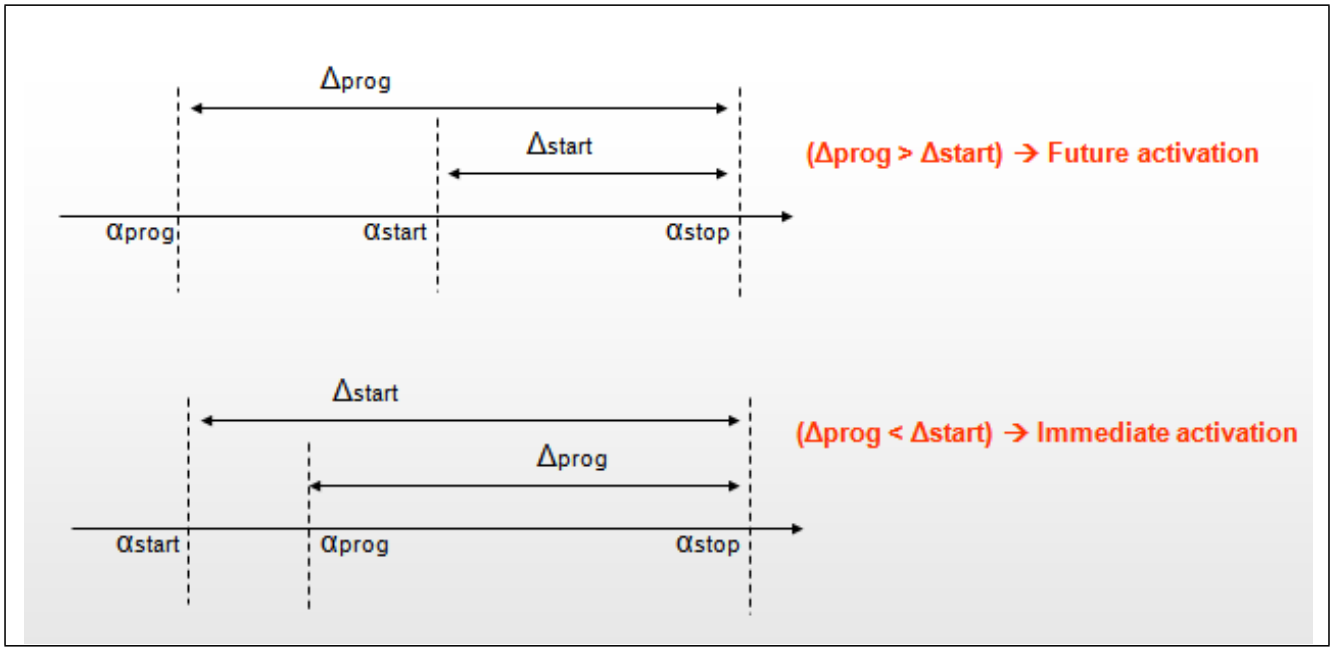

图 32 “过去/未来”应用策略

3.6.3 ATOM 中断

为了减少中断的次数,使用了中断集成器模块(ICM)将各个子模块的 GTM 中断线路捆绑成中断组。采用了该捆绑之后，GTM 外面可见的中断线路就减少了。

对于 ATOM 子模块，再次在 ICM 子模块内捆绑中断线路以减少外部的中断线路。中断是“ORed”形式的，指一个 GTM 外部中断线路代表了相邻的两个 ATOM 通道中断。

```
static void IfxMpg_initInterrupt(void) {
  /*SRPN= Service Request Priority Number (see SRC Register)*/
  SRC_GTMATOM00.U=(CPU0<<11)|(ENABLED_INT << 10)|ATOM00_SRPN;/*CH0-CH1*/
  SRC_GTMATOM01.U=(CPU0<<11)|(ENABLED_INT << 10)|ATOM01_SRPN;/*CH2-CH3*/
  SRC_GTMATOM02.U=(CPU0<<11)|(ENABLED_INT << 10)|ATOM02_SRPN;/*CH4-CH5*/
  SRC_GTMATOM03.U=(CPU0<<11)|(ENABLED_INT << 10)|ATOM03_SRPN;/*CH6-CH7*/


interruptHandlerInstall(ATOM00_SRPN,(uint32)&IfxMpg_atomProxyIsr);//CH0- CH1

interruptHandlerInstall(ATOM01_SRPN,(uint32)&IfxMpg_atomProxyIsr);//CH2- CH3

interruptHandlerInstall(ATOM02_SRPN,(uint32)&IfxMpg_atomProxyIsr);//CH4- CH5

interruptHandlerInstall(ATOM03_SRPN,(uint32)&IfxMpg_atomProxyIsr);//CH6- CH7
}
```

图 33 GTM 中断示例

4 实现样例-IfxMpg 驱动程序

4.1 简介

开发了一个基础的多功能脉冲产生(MPG)驱动程序作为“概念证明”。驱动程序的目的是表示如何使用 GTM 解决与点火和喷油脉冲管理相关的最重要的话题。

注意： 并未打算将该驱动程序用于实际产品，而是仅仅将其用作入门级开发的起点。

以下章节概述了驱动程序的 API(应用程序界面）及其用途。

如同驱动程序代码，还可以获得 doxygen 文档及 Windows 帮助文件。

4.2 驱动程序概览

MPG 驱动程序提供了一系列特性可用于产生基于时间或角度的并具备不同特征的脉冲。

驱动程序在一组“逻辑通道”上工作，这些逻辑通道可以产生不同种类的脉冲。每一个 MPG 通道都被映射到一个具体 ATOM HW 通道。

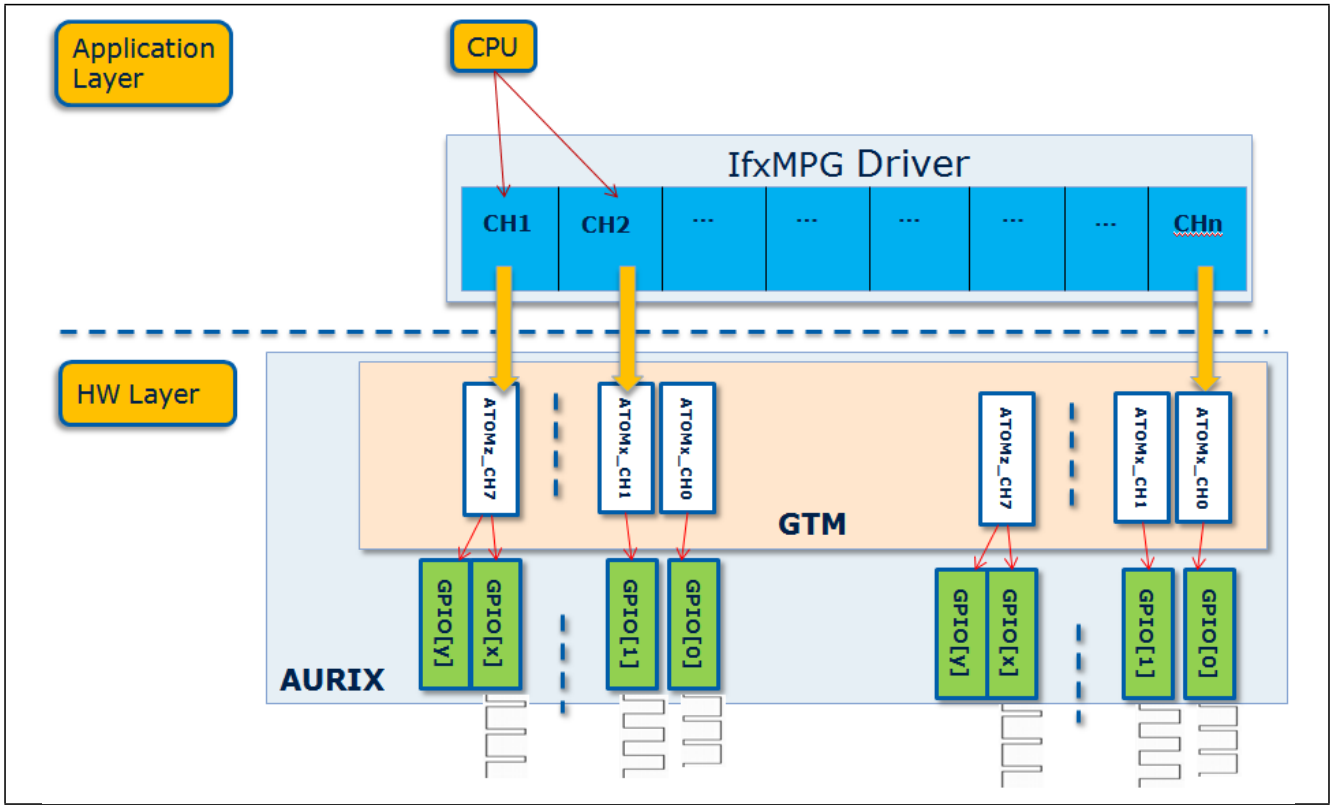

图 34 IfxMpg 驱动程序系统概览

通道所支持的脉冲类型如下:

•角度-角度
  -角度开始角度结束，用角度计数器的绝对值。
  -优先用于产生点火脉冲。
•角度-时间
  -角度（绝对值）开始，并在脉冲开始后一段定义的时间之后（绝对脉冲长度）结束。
  -优先用于产生点火脉冲，和直喷脉冲。
•时间-时间
  -时间开始时间结束，该时间指定时计数器的绝对值。
  -用于在固定时间模式下产生点火脉冲。
•时间-角度
  -终止角是一个绝对值，而开始角度的计算在脉冲长度内参照终止角。
  -用于产生普通的喷油脉冲。
•时间-时间脉冲群
  -一组脉冲群是通过开和关区间的绝对值来产生的。
  -用于产生点火和喷油脉冲群。
•角度-时间脉冲群
  -产生许多脉冲群。每个群都由若干可配置的 Ton / Toff 脉冲组成。
•角度-角度脉冲群
  -产生了一组脉冲群，其中它的每个脉冲都有各自可配置的开始和终止绝对角。

从软件的角度来看，每个通道都实现了一个基本的状态机可以持续跟踪脉冲产生的状态。这些状态指:

- ELAPSED:还未给脉冲设定。
- PROGRAMMED:已经给脉冲设置，但还未启动。
- IN_PROGRESS:脉冲正在处理中。
- CH_DISABLED: 通道禁止。

一般来说，当某个通道处于 ELAPSED 或者是 PROGRAMMED 状态时，并且收到一个新的请求，则前一个脉冲程序被重设。

当某个脉冲处于 IN_PROGRESS 状态，并且通道收到一个新的请求，那么前一个脉冲立即结束，并对新的脉冲设定。

驱动程序也可以产生异常（回调）以通知应用程序发生了与驱动程序的操作或脉冲的生成相关的特殊事件。所说的这些事件有:

•请求失败错误
•重叠错误
•脉冲开始
•饱和角度
•脉冲终止
•序列终止

4.3 公共函数-API

表 1 IfxMpg 驱动程序 API 函数

函数	签名	描述
IfxMpg	void **IfxMpg** (void)	驱动程序构造函数，初始化所有属性
IfxMpg_config	IfxMpg_PulseType	配置通道<chId> 以产生脉冲类型

实现示例-IfxMpg 驱动程序

功能	签名	描述
	IfxMpg_config(IfxMpg_IDNMpg chId, IfxMpg_PulseType pulseType, IfxMpg_Level ActiveLevel, IfxVrs_Mstatus* status)	
IfxMPG_getRealAngle	uint32 **IfxMPG_getRealAngle**(IfxMpg_IDNMpg chId, IfxVrs_Mstatus *status)	返回在通道<chId>上最后一次执行的脉冲的终止角
IfxMPG_getStartTime	uint32 **IfxMPG_getStartTime**(IfxMpg_IDNMpg chId, IfxVrs_Mstatus *status)	返回在通道<chId>上最后一次执行的脉冲的开始时间
IfxMPG_getRealTime	uint32 **IfxMPG_getRealTime**(IfxMpg_IDNMpg chId, IfxVrs_Mstatus *status)	返回在通道<chId>上最后一次执行的脉冲的长度
IfxMpg_getLevel	IfxVrs_Mstatus **IfxMpg_getLevel**(IfxMpg_IDNMpg chId, IfxVrs_Mstatus *status)	取得并返回通道输出引脚的当前实际电平
IfxMpg_getStatus	IfxVrs_Mstatus **IfxMpg_getStatus**(IfxMpg_IDNMpg chId, IfxVrs_Mstatus *status)	检索通道的状态(ELAPSED, PROGRAMMED, IN_PROGRESS, CH_DISABLED)
IfxMpg_programException	IfxVrs_Mstatus **IfxMpg_programException**(IfxMpg_IDNMpg chId, IfxMpg_Edge conditionEdge)	为通道<chId>编写异常触发程序
IfxMPG_angleAngle	IfxVrs_Mstatus **IfxMPG_angleAngle**(IfxMpg_IDNMpg chId, uint32 angStart, uint32 angStop)	产生一个脉冲，该脉冲在 reference =<AngStart> 时开始，在 reference = <AngStop>时终止
IfxMPG_angleTime	IfxVrs_Mstatus **IfxMPG_angleTime**(IfxMpg_IDNMpg chId, uint32 angStart, uint32 pulseLen, uint32 angSat)	产生一个脉冲，该脉冲在 reference =<AngStart> 时开始， 且在脉冲达到长度<pulseLen> 或者是达到饱和角度<AngSat>时终止
IfxMPG_angleAngleN	IfxVrs_Mstatus **IfxMPG_angleAngleN**(IfxMpg_IDNMpg chId, uint8 length, uint32* angStartArray, uint32* angStopArray);	编写产生 AngleAngleN 脉冲群的程序，该脉冲群包含了一组 <length>角度-角度脉冲。
IfxMpg_timeAngle	IfxVrs_Mstatus **IfxMpg_timeAngle**(IfxMpg_IDN Mpg chId, uint32 angStop, uint32 pulseLength,	这种类型的脉冲以角度开始，在某个时间结束。从脉冲的结束处(即脉冲基准)计算脉冲的开始。

实现示例-IfxMpg 驱动程序

功能	签名	描述
	uint32 angSat, IfxMpg_TimeAnglePrio priority)	为了保持脉冲长度的稳定,必须定期重新计算任意 TimeAngle 脉冲的开始角度。
IfxMpg_angleTimeN	IfxVrs_Mstatus **IfxMpg_angleTimeN**(IfxMpg_IDNMpg chId, uint8 length,uint8 nPulseN, uint32* startAnglesArray,IfxMpg_tOnOffPulsesMatrix* tOffArray, IfxMpg_tOnOffPulsesMatrix* tOnArray)	产生数组<length>脉冲群,每一个脉冲群都包含<nPulseN>脉冲。
IfxMpg_timeTime	IfxVrs_Mstatus **IfxMpg_timeTime**(IfxMpg_IDNMpg chId, uint32 relStopTime, uint32 pulseLength, uint32 maxAngAbs, uint32 minAngAbs, uint32 relativeAngle, IfxMpg_MpgPrio priority)	根据<priority>产生两种类型的时间-时间脉冲
IfxMpg_correctTime	IfxVrs_Mstatus **IfxMpg_correctTime** (IfxMpg_IDNMpg chId, uint32 pulseLength)	修改 AngleTime 或 TimeAngle 脉冲的持续时间
IfxMPG_timeTimeN	IfxVrs_Mstatus **IfxMPG_timeTimeN** (IfxMpg_IDNMpg chId, uint32 tOff, uint32 tOn, uint32 nPulse);	在规定的 Ton and TOff 时间(从 Toff 开始)产生一连串的<nPulse> 脉冲

下列代码片段是驱动程序应用示例:

```
// ****** Engine Position Driver Initialization
IfxVrs_init(NULL);
IfxVrs_acqConfig(IfxVrs_DiagnosticMode_noFault);
IfxVrs_setActiveEdge(IfxVrs_ActiveEdge_falling);
IfxVrs_setWaitStart(15000);        //Wait 15000xtCLK after the 1st tooth detected
IfxVrs_setNIgnoreTeeth(5, &status); //Wait 3 + [5] teeth in NO_SYNC
IfxVrs_setNMinPresyncTeeth(4, &status); //Wait 4 teeth is PRE-SYNC
IfxVrs_start(); //Start EP/VRS the driver

// ****** MPG Driver Initialization
IfxMpg();
IfxMpg_config(CH0, MPG_TIME_ANG, MPG_ACT_LEV_LOW, &status);
IfxMpg_config(CH1, MPG_ANG_ANG_N, MPG_ACT_LEV_HIGH, &status);
```

```
 IfxMpg_config(CH2, MPG_ANG_TIME_N, MPG_ACT_LEV_LOW, &status);
 IfxMpg_config(CH3, MPG_TIME_ANG, MPG_ACT_LEV_LOW, &status);
 IfxMpg_config(CH4, MPG_ANG_ANG, MPG_ACT_LEV_LOW, &status);
 IfxMpg_config(CH5, MPG_ANG_TIME, MPG_ACT_LEV_LOW, &status);
 IfxMpg_config(CH6, MPG_TIME_TIME_N, MPG_ACT_LEV_LOW, &status);
 IfxMpg_config(CH7, MPG_TIME_TIME, MPG_ACT_LEV_LOW, &status);
 IfxMpg_config(CH8, MPG_TIME_TIME, MPG_ACT_LEV_LOW, &status);

    //******* Wait for the DPLL/Flywheel sync.
  while (IfxVrs_getFlywheelStat() != IfxVrs_FlywheelStat_sync) ;

  while (1 == 1) {
    if (channelsArray[CH0].chStatus == ELAPSED) {
      /** PORT02.0*/
      pulseLenTimeAng = 5000; //5000us
      IfxMpg_timeAngle(CH0, 4000, pulseLenTimeAng, 6410, MPG_TIME_PRIO);
      __dsync();
    }
    if (channelsArray[CH1].chStatus == ELAPSED) {
      /** PORT02.1*/
      uint32 starAngArray[3] = { 250, 3600, 5400 };
      uint32 stopAngArray[3] = { 1500, 4200, 6600 };
      IfxMPG_angleAngleN(CH1, 3, &starAngArray, &stopAngArray);
      __dsync();
    }
    if (channelsArray[CH2].chStatus == ELAPSED) {
      /** PORT02.2*/
      uint32 starAngArray[3] = { 250, 3600, 5400 };
      uint32 stopAngArray[3] = { 1500, 4200, 6600 };
      IfxMpg_angleTimeN(CH2,3,4,&starAngArray,&tOffArrayGlob,
&tOnArrayGlob);
      __dsync();
    }
    if (channelsArray[CH3].chStatus == ELAPSED) {
      /** PORT02.3*/
      uint32 absAngStop = 5400; //540.0 degrees
      IfxMpg_timeAngle(CH3,          absAngStop,          pulseLenTimeAng,5410,
MPG_ANGLE_PRIO);
      __dsync();
    }
    if (channelsArray[CH4].chStatus == ELAPSED) {
      /** PORT02.4*/
      uint32 startAngAbs = 100; //10.0 degrees
      IfxMPG_angleAngle(CH4, 1800, 5400);
```

```
        //__dsync();
    }
    if (channelsArray[CH5].chStatus == ELAPSED) {
        /** PORT02.5*/
        uint32 startAngAbs = 1230; //123.0 degrees
        uint32 pulseLen = 5400; //5400us
        IfxMPG_angleTime(CH5, startAngAbs, pulseLen, 5400);
        __dsync();
    }
    if (channelsArray[CH6].chStatus == ELAPSED) {
        /** PORT20.0*/
        IfxMPG_timeTimeN(CH6, 5000, 4000, 5);
        __dsync();
    }
……
```

图 35　驱动程序应用示例

4.4 IfxMpg_timeTimeN

TimeTimeN 信号包含了一连串的脉冲。脉冲程序之后，立即产生一连串的脉冲，每一个脉冲都从<Toff> 时间开始，之后便是<Ton>时间。

有关编程时刻，遵循如下规则：

- 如果已经产生了前一个脉冲，则接受新的编程脉冲
- 如果还未开始前一个脉冲，则接受新的编程脉冲，前一个脉冲被覆盖
- 如果编程请求到达时，正在执行一个脉冲，那么立即停止前一个脉冲，并执行新的脉冲
- 如果<pulseNo>是 0，那么脉冲将继续产生一系列 Ton，Toff 信号

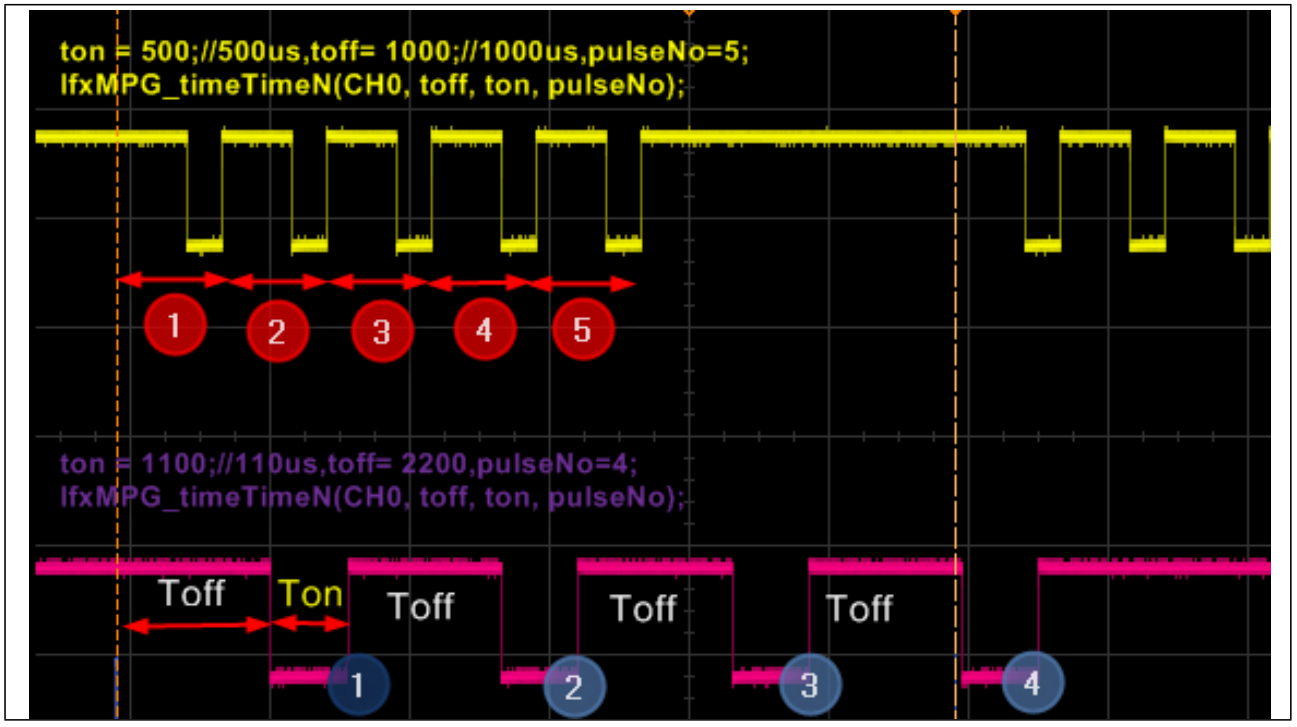

图 36 时间-时间 N：图解

4.5 IfxMpg_angleAngle

角度-角度脉冲以角度开始，并以角度结束(<AngStart>,<AngStop>)。

以下规则适用于脉冲开始瞬间：

•如果已经产生了前一个脉冲(ELAPSED)，那么接受新的脉冲编程
•如果还未开始前一个脉冲，则接受新的编程脉冲，前一个脉冲被覆盖
•如果当编程请求到达时，正在执行一个脉冲，那么立即停止前一个脉冲，并给新的脉冲编程

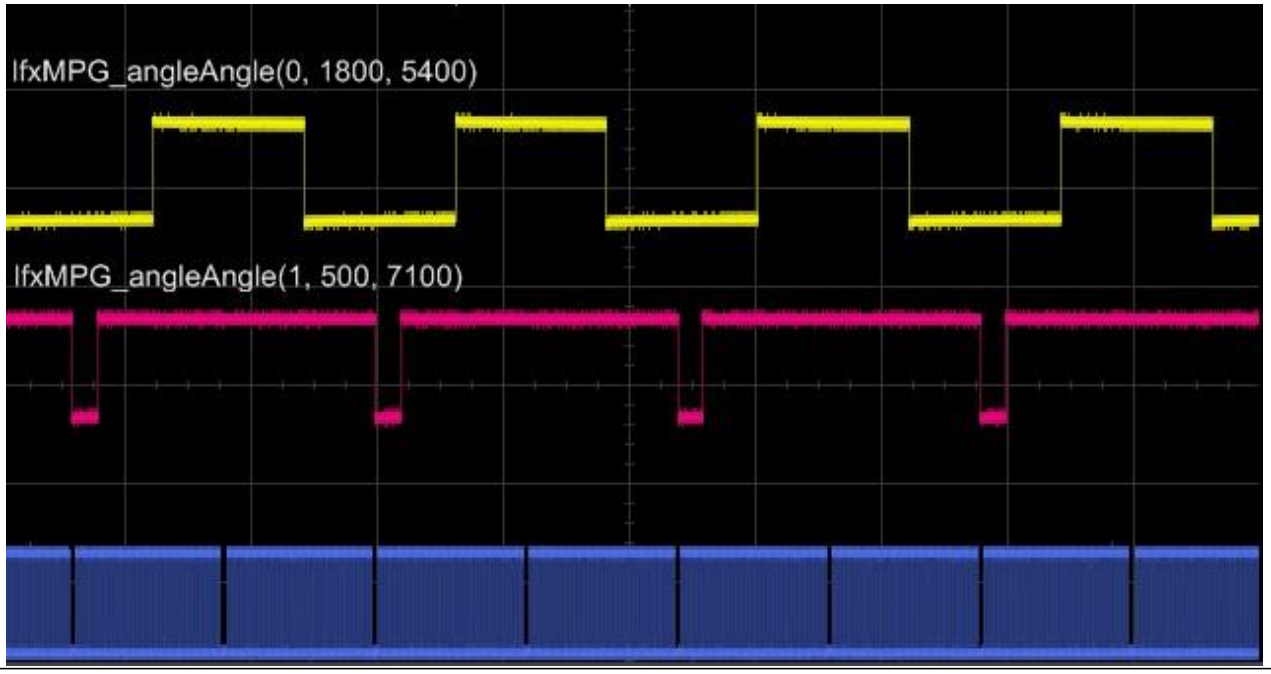

图 37 角度-角度脉冲图

4.6 IfxMpg_angleAngleN

AngleAngleN 脉冲包含了一连串的脉冲,其中每个脉冲都有自己可配置的绝对开始角和终止角。遵循如下规则:

•如果已经产生了前一个脉冲组，那么接受新的编程脉冲
•如果还未开始前一个脉冲组，那么接受新的编程脉冲，前一个脉冲被覆盖
•如果当编程请求到达时，正在执行一个脉冲，那么立即停止前一个脉冲，并给新的脉冲编程

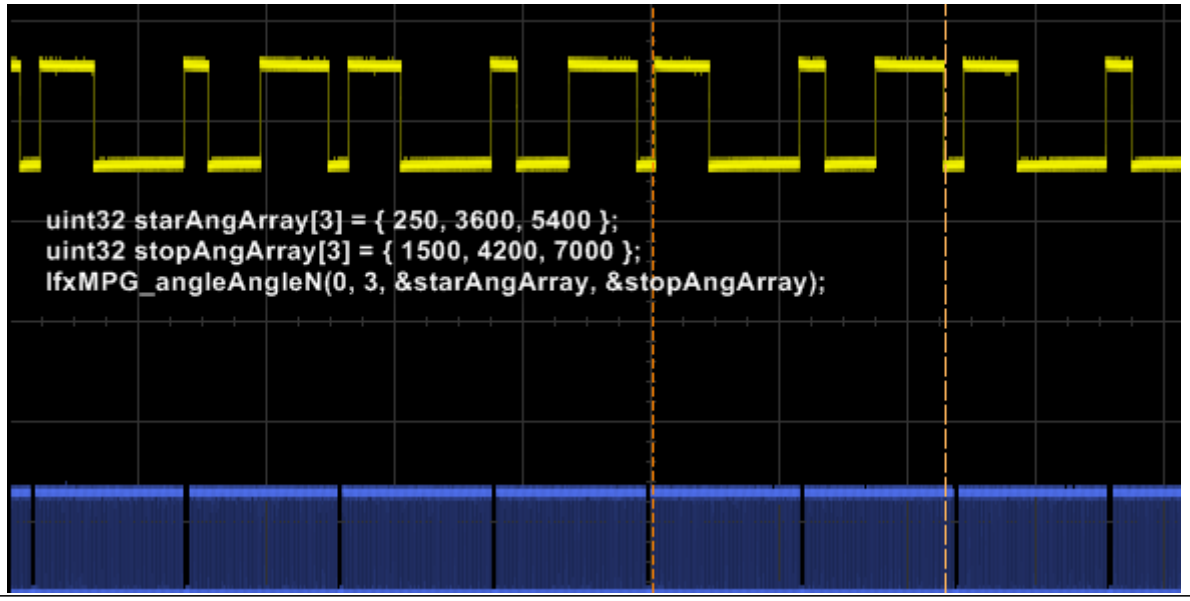

图 38 Angle-AngleN 脉冲图

4.7 IfxMpg_angleTime

角度-时间脉冲以绝对角被激活(<AngStart>),并在一定的时间(由脉冲长度定义)过去后结束,或者是首先达到饱和角度<AngSat>时结束(仅在<AngSat>!=0xFFFF 时)。

角度-时间脉冲遵循如下要求:

- 如果已经产生了前一个脉冲，那么接受新的编程脉冲
- 如果还未开始前一个脉冲，那么接受新的编程脉冲，前一个脉冲被覆盖
- 如果当编程请求到达时，正在执行一个脉冲，那么立即停止前一个脉冲，并给新的脉冲编程

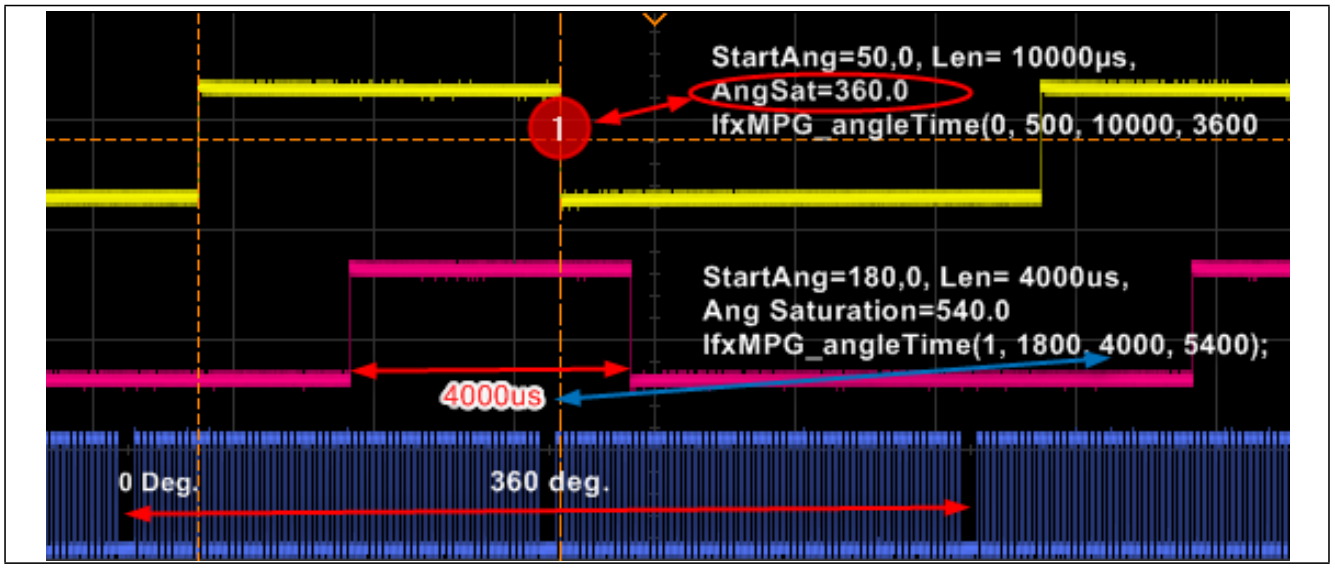

图 39 Angle-Time 脉冲图

4.8 IfxMpg_timeAngle

TimeAngle 脉冲在特定的角度(<AngStop>)结束,且编程保持最新,以便脉冲在(<AngStop>)角度值之前,在一段规定的时间(<PulseLen>) 内保持有效(打开)。定期重新计算开始角度（即使是在出现加速/减速时）以便保持脉冲长度稳定。该脉冲类型总是“以角度开始”，但根据所选的优先级，脉冲结束情况不尽相同：

- TIMO_PRIO
  -脉冲在时间里关闭。如果饱和角在脉冲长度消逝之前达到，最终脉冲以饱和角度结束(如果<AngSat>!=0xFFFF)。
- ANGLE_PRIO
  -脉冲总是以角度(<AngStop>)结束。因此饱和角度被忽略。在该模式下，如果有加速/减速，在脉冲开始后，脉宽长度可被缩短或加长。

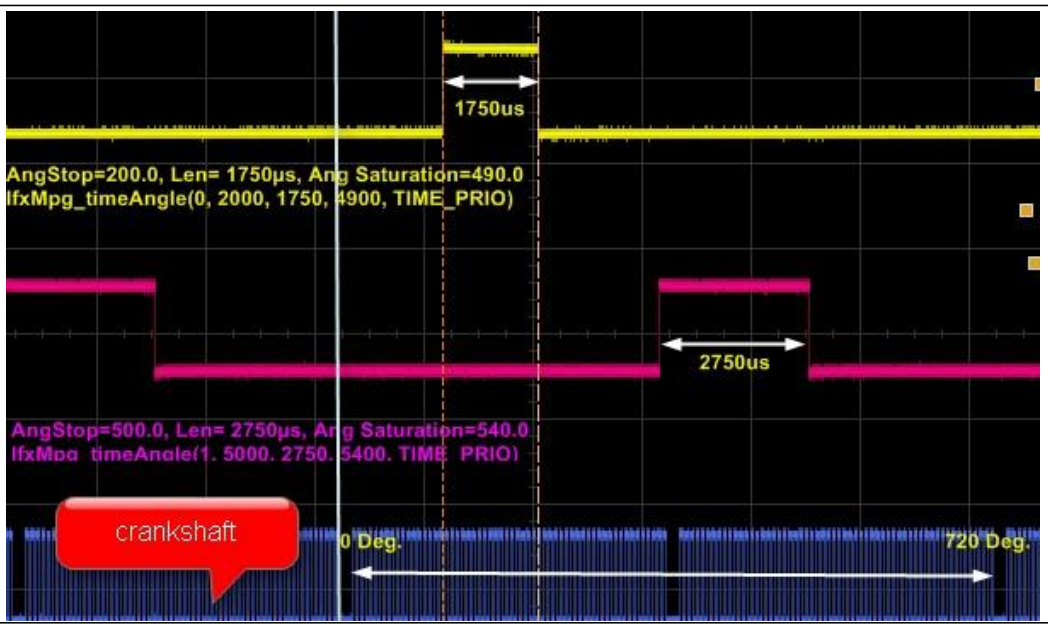

图 40 时间-角度图解：TIME PRIO, AngSat > TStop

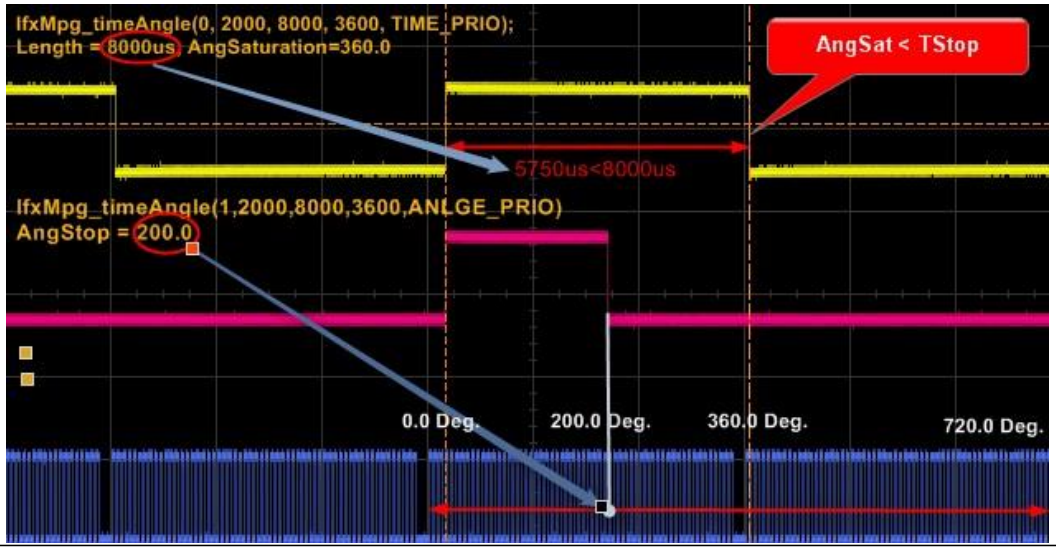

图 41 时间-角度：TIME PRIO versus ANGLE_PRIO

4.9 IfxMpg_angleTimeN

Angle-TimeN 信号包含了 N 串脉冲,每串包含了 M 个子脉冲。

每一串脉冲，其起始脉冲（以<StartAng[N]>定义的角度）在<Ton> 时间内保持有效，紧跟其后的一段脉冲在<Toff>时间内保持无效。

在 Ton[N,max(Npulse)] and Toff[N, max(Npulse)-1]矩阵里配置每个脉冲的 Ton / Toff 时间。

遵循如下规则：

•如果已经产生了前一个脉冲组，那么接受新的脉冲编程
•如果还未开始前一个脉冲组，那么接受新的编程脉冲，前一个脉冲覆盖
•如果当编程请求到达时，正在执行一个脉冲，那么立即停止前一个脉冲，并给新的脉冲编程

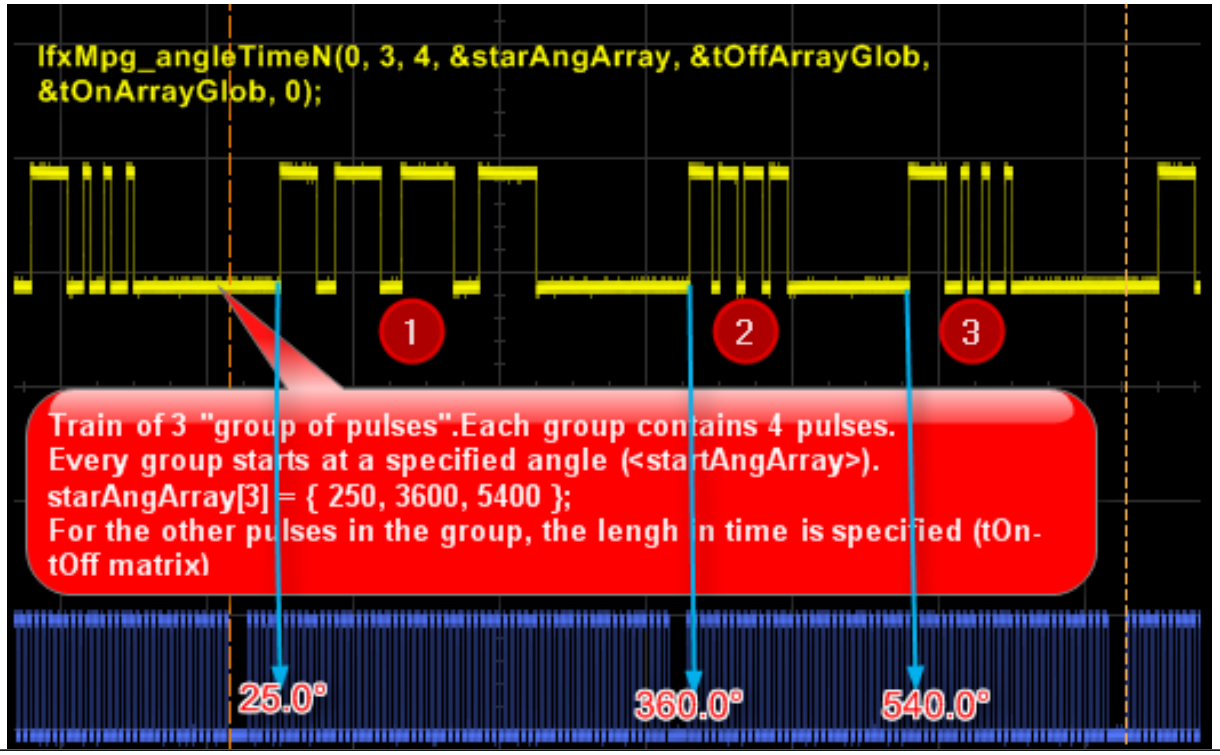

图 42　图 1 Angle-Time N 图解

4.10 IfxMpg_timeTime

TimeTime 脉冲在从脉冲设定开始的一段时间耗尽后被激活。而在脉冲长度消逝后被关闭。该脉冲必须遵守所有的角度限制（最小和最大角度范围）。

- NO_PRIO_DIFF
  -脉冲在时间里开始。通过绝对终止时间(<StopTime>)和脉冲长度(<PulseLength>)计算出开始时间。为了启动补偿，以便脉冲正确参照相对角度(<RelativeAngle>)，需要进行一组计算（见优先级低的脉冲长度）。
- PRIO_DIFF
  -该脉冲的生成使用了一定的机制以便将脉冲长度区分优先次序。这要求定期重新计算终止角，并在 MinAng- MaxAng 间隔里检查终止角。根据终止角的变化也要重新计算开始角，并保持脉冲宽度。（见优先级较高的 PulseLen）。

图 43 TimeTime PrioDiff 图解

w ww.infineon.com

Published by Infineon Technologies AG