AN12732

HDMI-CEC Implementation on LPC55xx

Rev. 0 — 25 February 2020 Application Note

Contents

1 Introduction.....1
1.1 Glossary..... 1
2 HDMI-CEC.....2
2.1 Overview.....2
2.2 Bit-level protocol.....2
2.3 Block-level protocol..... 4
2.4 Frame-level protocol.....5
2.5 Device connectivity and addressing..... 5
2.6 CEC message descriptions.....5
3 Implementation.....6
3.1 Overview.....6
3.2 Hardware connection....7
3.3 Timer Capture function..... 8
4 Example and Test.....11
5 Conclusion.....12

1 Introduction

This application note describes how to implement High-Definition Multimedia Interface (HDMI)-Consumer Electronics Control (CEC) functions as a TV set or a projector that support the HDMI-CEC protocol based on LPC5500 series with GPIO and SCT used.

CEC is a single-wire bus protocol allowing AV products to discover and communicate with one another across a system. CEC makes possible global controls, therefore minimize the number of IR remotes and keypress required for basic operation of a system. It is specified to operate at low speeds with minimum processing and memory overhead.

In general, the CEC bus allows all products in the system to potentially discover and communicate with each other. Also, it enables global-simplified (single remote) system control in HDMI-interfaced systems.

For more details, see the high-definition multimedia interface specification (www.hdmi.org).

This application note does not include high-level description of HDMI-CEC protocol but focuses on the physical level implementation on MCU.

1.1 Glossary

Table 1. Abbreviation

Items	Description
HDMI	High-Definition Multimedia Interface
(HDMI) Source	A device with an HDMI output
(HDMI) Sink	A device with an HDMI input
Initiator	The device, which is sending, or has just sent a CEC message and, if appropriate, is waiting for a follower to respond
Follower	A device which has just received a CEC message and is required to respond to it
CEC	Consumer Electronics Control
Broadcast Message	A message sent to logical address 15, that all devices are expected to receive
TV	A device with an HDMI input which has the ability to display the input HDMI signal. Generally, it has no HDMI output.

Table continues on the next page...

Table 1. Abbreviation (continued)

Destination	The target device for a CEC message
Source Device	The device which is currently providing an AV stream via HDMI
Logical address	A unique address assigned to each device

2 HDMI-CEC

2.1 Overview

The CEC bus is a one-wire, “party line” protocol that connects up to ten (10) AV devices through standard HDMI cables. The CEC protocol includes automatic mechanisms for physical address (topology) discovery, (product-based) logical addressing, arbitration, retrans-mission, broadcasting, and routing control. A typical application of HDMI system is shown in Figure 1, where all the HDMI devices are connected together through the HDMI cables serially. The HDMI-CEC bus connects up to 10 HDMI AV devices.

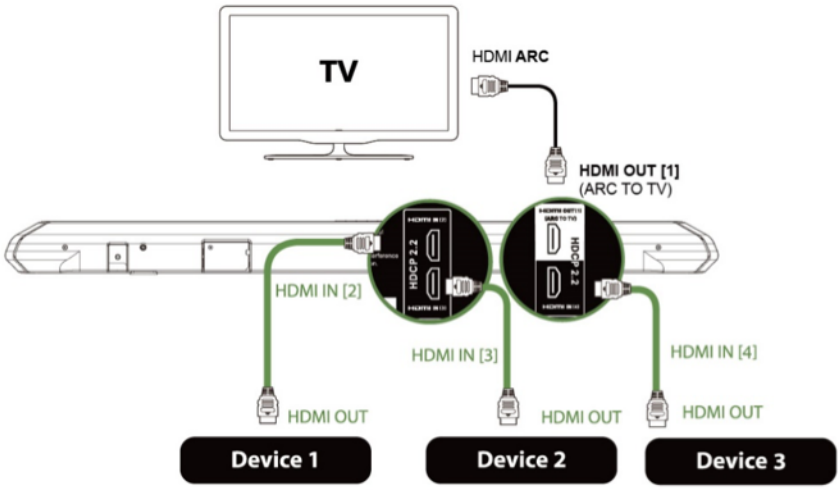

Figure 1. HDMI-CEC system

2.2 Bit-level protocol

Communication is always between an initiator and one (or more) follower(s). Both ini-tiator and follower(s) can assert bits. The initiator-asserted bits provide data, while the follower-initiated bits provide acknowledgment. Bit-level communication rates is less than 500 bits/second. The messages begin with one long start bit and are immediately followed by a number of shorter data bits. When CEC bus is idle, the bus level is always pulled to VCC.

2.2.1 Start Bit timing

The start bit pulse format is shown in Figure 2. It identifies the start of a new frame, a low-pulse plus a high-pulse within a certain validated duration.

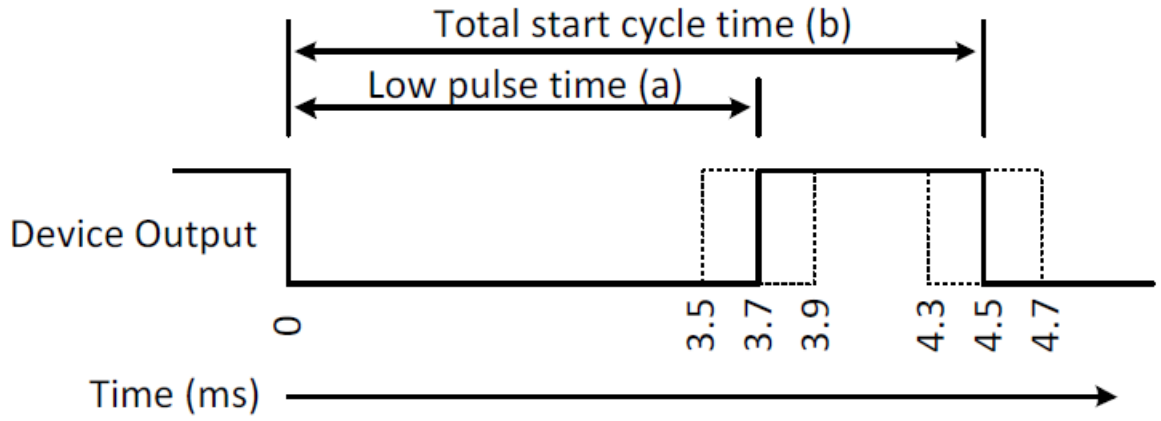

Figure 2. Start Bit Timing

2.2.2 Data Bit timing

The data bit is followed by start bit, each data bit is consisting of a low duration and a high duration. The high to low transition at the end of data bit is the start of next data bit and only occurs if there is a following data bit. After transmitting the final bit, the CEC line remains high as shown in Figure 3.

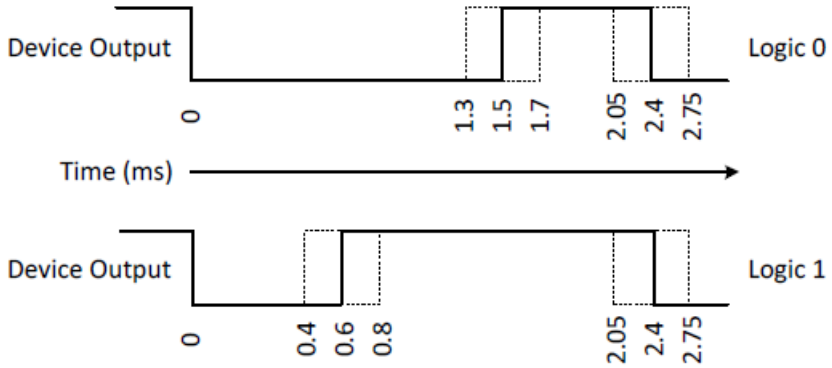

Figure 3. Data Bit timing

2.2.3 Acknowledgment Bit timing

Acknowledgment bit is also called asserted bit, the data bit is sent by initiator, and the follower where the follower may assert the bit to logical 0 to acknowledge a data block. The initiator outputs a logical 1, therefore allowing the follower to change the CEC state by pulling the line during the safe sample period and making the Acknowledge (ACK) bit from logical 1 to logical 0. So, it is the follower that controls the acknowledgment bit‘s logical level, as shown in Figure 4.

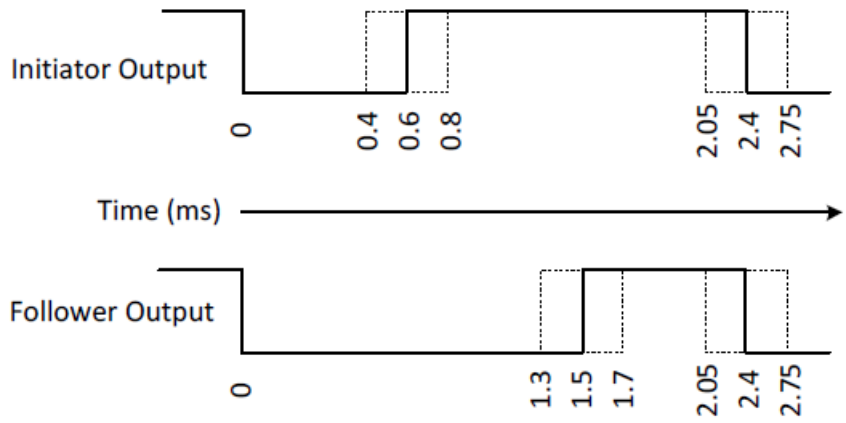

Figure 4. Acknowledgment Bit Timing

2.3 Block-level protocol

Bits are grouped into 10-bit header and data blocks. Both header and data blocks include 8-bits of data along with End of Message (EOM) and ACK bits. The EOM bit signals the final block in a message. A '0' indicates that one or more blocks follow and a '1' indicates that the message is complete.

Data Block									
7	6	5	4	3	2	1	0	--	--
Informational Bits								EOM	ACK

Figure 5. Data block

EOM:

- 0: one or more data blocks follow.
- 1: the message is complete, no more data.

ACK:

- 0: follower asked for the message.
- 1: no acknowledgment.

The header block is identical to data block. It is followed by start bit immediately as the first "data block" in a message. The header block's 8 data bit indicates the initiator address and destination address, as shown in Figure 6.

Header Block									
3	2	1	0	3	2	1	0	--	--
Initiator Address				Destination Address				EOM	ACK

Figure 6. Header block

2.4 Frame-level protocol

HDMI CEC messages are sent using frames. Each CEC frame consists of a start bit, a header block, and possibly data blocks. An example of HDMI-CEC frame is shown in Figure 7.

Figure 7. HDMI-CEC frame

The first data block followed by header block called opcode, then followed by zero, one or many data blocks indicating the actual data transferred.

2.5 Device connectivity and addressing

To allow CEC to address specific physical devices and control switches, all devices shall have a physical address. This connectivity must be worked out whenever a new device is added to the system. The physical address discovery process uses only DDC/EDID mechanism and can be applied to all HDMI sinks and sources.

The CEC line is directly connected to all nodes on the network. After discovering their own physical address, the CEC device transmits their physical and logical address to all other devices, therefore, allows the system to create a map of netwDork.

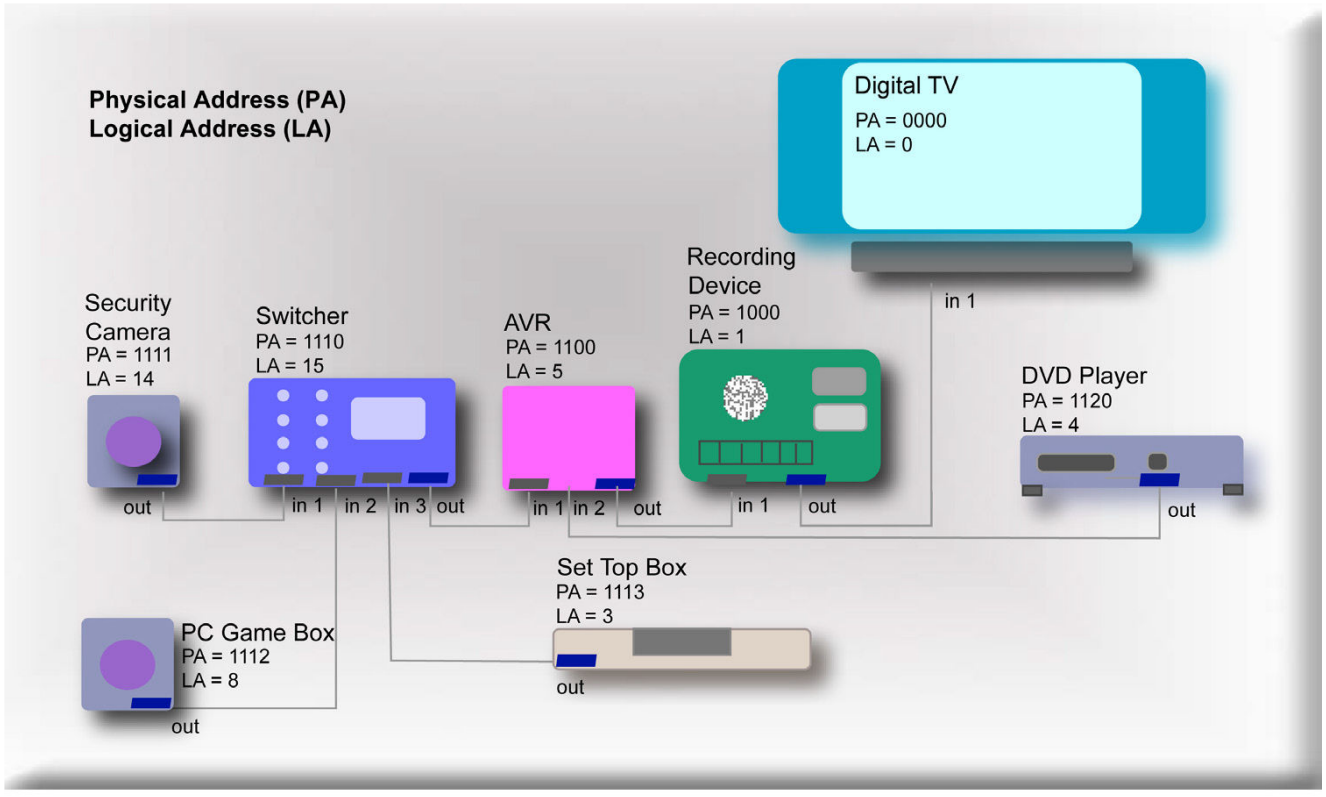

Figure 8. Physical Address within a HDMI system

2.6 CEC message descriptions

Table 2 shows the most common used commands in CEC message. For more opcodes, parameters and description or the other message, see the HDMI-CEC specification.

Table 2. CEC message description

Opcode	Value	Description	Parameters	Response
<Polling Message>	-	Used by any device for device discovery	None	Set a low-level Ack
<Give Physical Address>	0x83	A request to a device to return its physical address	None	Report physical address
<Report Physical address>	0x84	Used to inform all other devices of the mapping between physical and logical addresses of the initiator	Physical address and device type	-
<Active Source>	0x82	Used by a new source to indicate that it has started to transmit a stream or used in response to a <Request Active Source>	[Physical Address]	A current active source should take appropriate action. TV should switch to the appropriate input. Any CEC switches to the appropriate input and comes out of standby if necessary
<Image View On>	0x04	Sent by a source device to the TV whenever it enters the active state (alternatively, it may send <Text View On>)	None	Turn on (if not on). If in 'Text Display' state, the TV enters 'Image Display' state
<Give Device Power Status>	0x8F	Used to determine the current power status of a target device	None	<Report Power Status>
<Report Power Status>	0x90	Used to inform a requesting device of the current power status	[Power Status]	-

3 Implementation

3.1 Overview

This section describes how to use LPC5500 series to implement low-level CEC bus protocol. The LPC55S6x/LPC55S2x/LPC552x is an Arm Cortex®-M33-based microcontroller for embedded applications. These devices include up to 320 kB of on-chip SRAM and up to 640 kB on-chip flash. It has SCT timer that can be used to capture external bus edge changes. We use this feature to capture raising and falling time in CEC bus.

Table 3. MCU peripheral resource used

IP used	Description
SCT	Capture edge timestamp
GPIO	Output and input CEC data
UART	Display log

3.2 Hardware connection

To demonstrate the CEC functions on LPC500 series, a Chromecast is used as initiator and an HDMI splitter is used to develop the system hardware. The system structure for the demonstration is shown in Figure 9. Note that the MCU emulates TV's CEC controller.

The HDMI-CEC pin is connected to two pins on LPC5500 series, one is used as GPIO output and the other is used as GPIO input and SCT input. A default debug console in SDK is also enabled for log display.

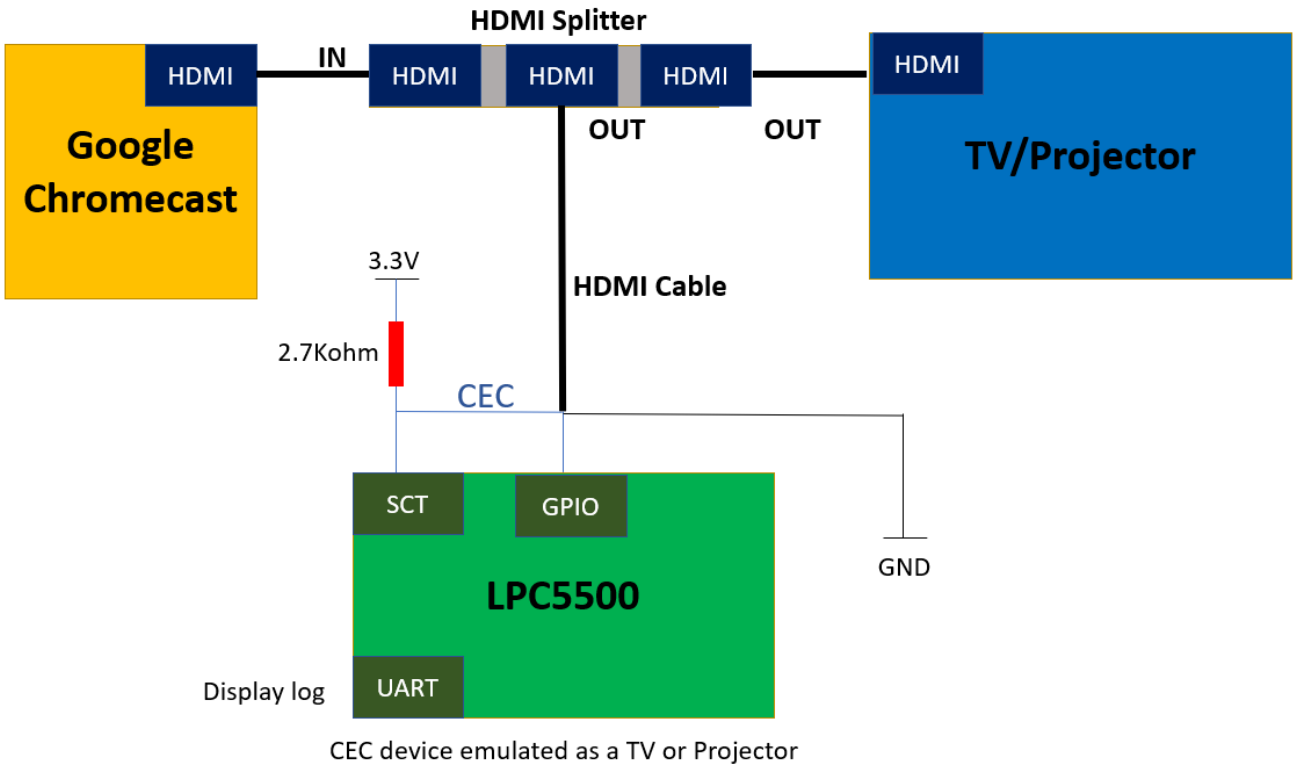

Figure 9. Physical address within an HDMI system

Figure 10 shows CEC pin and GND pin definition in HDMI connector.

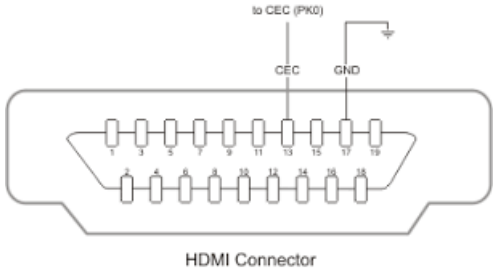

Figure 10. CEC pin and GND definition in HDMI connector

3.3 Timer Capture function

The most important function implemented in CEC protocol is input capture feature. This feature can be implemented by SCT module.

The State Configurable Timer (SCTimer/PWM) is a peripheral that is unique to NXP Semiconductors. It can operate like most traditional timers, but also adds a state machine to give it a higher degree of configurability and control. This allows the SCT to be configured as multiple PWMs, a PWM with dead-time control, and a PWM with reset capability, in addition to many other configurations that cannot be duplicated with traditional timers. Once the SCTimer/PWM has been configured, it can run autonomously from the microcontroller core, unless the SCTimer/PWM interrupt has been enabled which requires the core to service the interrupt.

This application note describes how to configure the SCT into a simple usage for input capture, which calculates the time span between two edge changes. Three SCT event registers, two capture registers, and one match register are used, as shown in the following table.

Table 4. SCT resources used

SCT resources	Description
EVT0	Configured as rising edge trigger
EVT1	Configured as falling edge trigger
EVT2	Configured as match trigger, used as timeout control
MAT0	Used for timeout counting
CAP0	Works with EVT0 to capture rising edge time
CAP1	Works with EVT1 to capture falling edge time

Before configuring SCT resource registers, capturing and matching register in SCT, SCT, and set clock diver need to be initialzed. Follow the steps below:

1. The SCT input clock is from BusClock, which is 150 MHz. Divide it by 150, so SCT clock is 150 Mhz/150 = 1 MHz.

```
uint32_t  sctimerClock = CLOCK_GetFreq(kCLOCK_BusClk);

    CLOCK_EnableClock(kCLOCK_Sct0);
    CLOCK_EnableClock(kCLOCK_InputMux);

    INPUTMUX->SCT0_INMUX[SCT_GPI0] = 7;

    SCT0->CONFIG |= SCT_CONFIG_UNIFY_MASK;

    /* config presclar
```

```
  */SCT0->CTRL &= ~SCT_CTRL_PRE_L_MASK;
                   SCT0->CTRL |= SCT_CTRL_PRE_L(149);
```

2. Configure SCT match and event register. MAT0 is used for timeout time setting, in this demo code.
3. Set the match register to 10000 which equals to 1M/10000 = 100 ms.
4. Set EVT0 as input rising edge trigger and EVT1 as input falling edge trigger.
5. Set EVT2 as match trigger, when SCT timer reaches MAT0 value. EVT2 is triggered and is used as timeout event flag.

```
SCT0->MATCH[MAT_INDEX] = 10000;
SCT0->MATCHREL[MAT_INDEX] = 10000;

/* EVT0 config */
SCT0->EV[EVT0_INDEX].CTRL = SCT_EV_CTRL_MATCHSEL(MAT_INDEX) | SCT_EV_CTRL_COMBMODE(2) |
SCT_EV_CTRL_IOCOND(1) | SCT_EV_CTRL_IOSEL(SCT_GPI0); /* raising */
SCT0->EV[EVT0_INDEX].STATE = (1 << 0);

/* EVT1 config */
SCT0->EV[EVT1_INDEX].CTRL = SCT_EV_CTRL_MATCHSEL(MAT_INDEX) | SCT_EV_CTRL_COMBMODE(2) |
SCT_EV_CTRL_IOCOND(2) | SCT_EV_CTRL_IOSEL(SCT_GPI0);    /* falling */
SCT0->EV[EVT1_INDEX].STATE = (1 << 0);

/* EVT2 */
SCT0->EV[EVT2_INDEX].CTRL = SCT_EV_CTRL_MATCHSEL(MAT_INDEX) | SCT_EV_CTRL_COMBMODE(1) |
SCT_EV_CTRL_IOCOND(2) | SCT_EV_CTRL_IOSEL(SCT_GPI0);    /* ti */
SCT0->EV[EVT2_INDEX].STATE = (1 << 0);
```

Finally, perform the below miscellaneous settings:

1. Configure CAP0 and CAP1 as input capture function.
2. When EVT0 and EVT1 are triggered, reset SCT timer.
3. Start SCT timer.

NOTE

In this application, SCT state is always 0; SCT state function is not used.

```
/* use as cap */
SCT0->REGMODE = (1<<CAP0_INDEX) | (1<<CAP1_INDEX);
SCT0->CAPCTRL[CAP0_INDEX] = (1<<EVT0_INDEX);
SCT0->CAPCTRL[CAP1_INDEX] = (1<<EVT1_INDEX);
/* Reset Counter L when Counter L event occurs */SCT0->LIMIT |=
        SCT_LIMIT_LIMMSK_L((1<<EVT0_INDEX) | (1<<EVT1_INDEX));
/* Start the L counter */
SCT0->CTRL &= ~SCT_CTRL_HALT_L_MASK;
```

4. Create a wait_lv function used to return the time between two edge changes:

```
static uint32_t wait_lv(uint8_t lv)
{
    int cnt = 0;
    cnt = sct_wait_lv(lv) / 100;
```

```
    return cnt;
}
```

The CEC bit implementation API is as follows:

```
static uint8_t   get_bit(uint8_t *val)
{
 uint32_t dt0, dt1;
 uint32_t ret = 0;
 dt0 = wait_lv(0);
 dt1 = wait_lv(1);
 if(abs(dt0 - CEC_TIMING_BIT0_LOW) < CEC_TIMING_TOR && abs(dt1 - CEC_TIMING_BIT0_HIGH) <
CEC_TIMING_TOR)
  {
   *val = 0;
    ret = 0;
  }
 else if(abs(dt0 - CEC_TIMING_BIT1_LOW) < CEC_TIMING_TOR && abs(dt1 - CEC_TIMING_BIT1_HIGH) <
CEC_TIMING_TOR)
  {
    *val = 1;
     ret = 0;
  }
else
 {
 ret = 1;
 }
  return ret;
}
static void
        set_bit(uint8_t val)
{
if(val)
  {
        pin_write(0);
        DelayUs(CEC_TIMING_BIT1_LOW*100);
        pin_write(1);
        DelayUs(CEC_TIMING_BIT1_HIGH*100);
  }
else
{       pin_write(0);
        DelayUs(CEC_TIMING_BIT0_LOW*100);
        pin_write(1);
        DelayUs(CEC_TIMING_BIT0_HIGH*100);
}
}
static uint32_t get_start_bit(void)
{
     uint32_t dt0, dt1;

     dt0 = wait_lv(0);
     dt0 = wait_lv(0);
     dt1 = wait_lv(1);
     if(abs(dt0 - CEC_TIMING_START_LOW) < CEC_TIMING_TOR && abs(dt1 - CEC_TIMING_START_HIGH) <
CEC_TIMING_TOR)
  {
      return 0;
  }
```

```
        return 1;
}
```

4 Example and Test

In this demo, the Google Chromecast acts as initiator, when there is no other CEC device on the bus. The Chromecast polls bus by sending header blocks several times as shown in Figure 11.

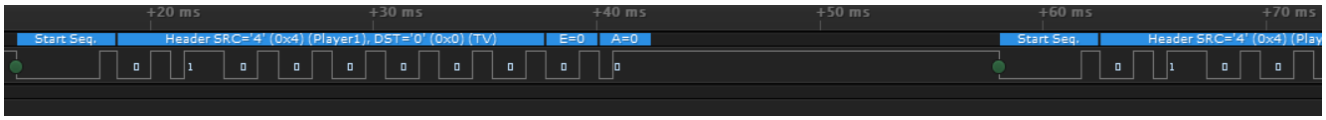

Figure 11. Physical Address within a HDMI system

The default CEC address for MCU is 0x00. When MCU connects to CEC bus, it responds to the destination address 0x00. Then, the Chromecast detects that the ACK bit is pull down and sends the following data block, as shown in Figure 12.

Figure 12. Chromecast send: GiveDevicePowerStatus(0x8F) frame

Then, MCU receives opcode, GiveDevicePowerStatus, and sends acknowledgment frame. This finishes a complete CEC bus transition, as shown in Figure 13.

Figure 13. MCU Responds: Report Power Status(0x90) frame

MCU monitors CEC bus activity and displays log on UART:

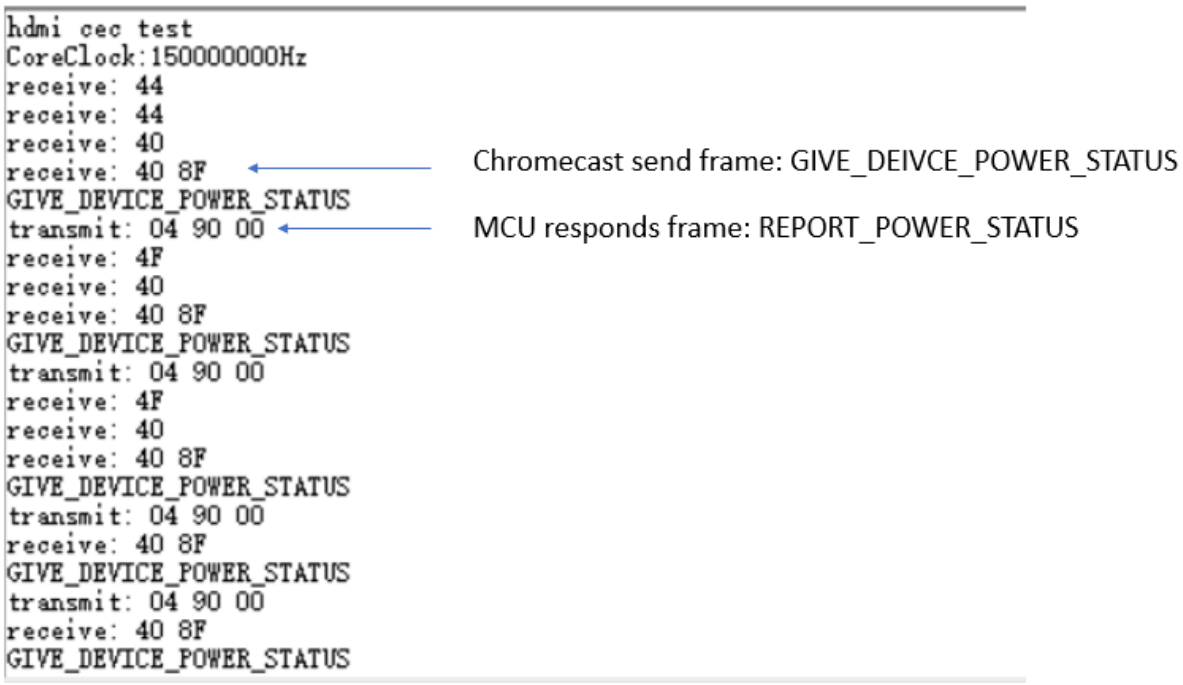

Figure 14. MCU UART log

5 Conclusion

This AN describes how to implement HDMI-CEC low-level protocol on LC5500 series. It also provides example code and instructions on hardware and software setup. It introduces the basic usage of SCT, a versatile state timer unique to NXP Semiconductor.

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer’s applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 25 February 2020

Document identifier: AN12732

arm